UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN DE UNA ASOCIACIÓN SCOUT

Miguel Herrera Carrillo
Tutor: Miguel Ángel Mora

Julio 2020

DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN DE UNA ASOCIACIÓN SCOUT

AUTOR: Miguel Herrera Carrillo
TUTOR: Miguel Ángel Mora

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2020

Resumen (castellano)

Las asociaciones de grupos scout han existido desde principios del pasado siglo, siendo su objetivo principal el proporcionar a jóvenes de diverso rango de edad una educación no formal basada en valores como el trabajo en equipo, la convivencia, el respeto, y el contacto con la naturaleza.

Concretamente, este proyecto está enfocado para el grupo scout Estrella Polar, perteneciente al distrito de Ciudad Lineal de Madrid.

Las principales tareas que se llevan a cabo en dicha asociación consisten en la realización y preparación de actividades y campamentos, así como la gestión tanto administrativa como económica del grupo, todo ello realizado por jóvenes voluntarios.

Este Trabajo de Fin de Grado consiste en el desarrollo de una aplicación web que sirva como plataforma electrónica para el grupo scout, donde tanto miembros del grupo, como sus familiares, y, sobre todo, el equipo de monitores, puedan tener un soporte electrónico unificado en el que recopilar toda la información necesaria para la gestión del grupo scout.

Por lo tanto, la aplicación está dirigida a tres tipos de usuarios distintos: scouts, familiares, y monitores, siendo este último perfil el más importante, puesto que la mayor parte de la funcionalidad de la aplicación está enfocada a la administración del grupo.

A grandes rasgos, esta aplicación permite funcionalidades tales como la gestión de inventarios de material y tiendas de campaña del grupo, el uso de un calendario donde publicar las diversas actividades que se van realizando, una sección de encuestas para poder conocer la opinión de los distintos tipos de usuario sobre diversos temas, y, por último, varias secciones donde se recopila distinta información de acampadas, rutas, objetos perdidos y contenido multimedia del grupo scout.

Adicionalmente, también se permite la personalización del perfil del usuario, y hay varias posibilidades de interacción entre los mismos, proporcionando así un enfoque de red social.

Se trata de una aplicación web progresiva (PWA), implementada usando Nuxt.js como Frontend, que se trata de un entorno de desarrollo basado en JavaScript, concretamente en Vue.js y Node.js.

En cuanto al Backend, se ha escogido la plataforma Firebase, proporcionada por Google.

Palabras clave (castellano)

Grupo Scout, Estrella Polar, Aplicación Web Progresiva, Nuxt.js, Vue.js, Node.js, JavaScript, Firebase.

Abstract (English)

The Scout movement has been around since the beginning of the last century, being its goal to provide young people a non-formal education based in values such as teamwork, coexistence, respect and contact with the nature.

Specifically, this project is focused on the Estrella Polar scout group, located in the Ciudad Lineal district of Madrid.

The main tasks carried out in this association consist of carrying out and planning activities and camps, as well as the administrative and economic management of the group, all carried out by young volunteers.

This Bachelor Thesis consists of the development of a web application that serves as an electronic platform for the scout group, where both members of the group, as well as their relatives, and, above all, the team of managers, can have an unified electronic support to collect all the information necessary for the management of the scout group.

Therefore, the application is aimed at three different types of users: scouts, family members, and monitors, the latter profile being the most important, since most of the application's functionality is focused on the group's administration.

Broadly speaking, this application allows functionalities such as the management of material inventories and group tents, the use of a calendar to publish the various activities that are carried out, a section of polls to be able to know the opinion of the different user types on various topics, and, finally, various sections where different information on camping, routes, lost objects and multimedia content of the scout group is collected.

Additionally, personalization of the user's profile is also allowed, with several possibilities of interaction between the users, thus providing a social network approach.

It is a progressive web application (PWA), implemented using Nuxt.js as Frontend, which is a framework based on JavaScript, specifically in Vue.js and Node.js.

For the Backend, the Firebase platform is used, provided by Google.

Keywords (inglés)

Scout Group, Estrella Polar, Progressive Web Application, Nuxt.js, Vue.js, Node.js, JavaScript, Firebase.

Agradecimientos

En primer lugar, agradecer al conjunto del equipo docente del Grado en Ingeniería Informática de la Universidad Autónoma de Madrid, por la excelente calidad de enseñanza a lo largo de los pasados cinco años, y, en concreto, al tutor de este proyecto, Miguel Ángel Mora, por solucionar y aclarar todas las diversas dudas que se han ido teniendo a lo largo del transcurso del TFG.

Al conjunto de miembros del Grupo Scout Estrella Polar, por ayudarme con el desarrollo de la aplicación y proponerme ideas útiles con las que perfeccionar la misma.

A mis compañeros de la carrera, por trabajar y apoyarnos durante el desarrollo de las distintas prácticas realizadas a lo largo de los últimos años en el grado.

Por último, a mi familia, por confiar en mí y ayudarme durante mi estancia en la universidad, proporcionándome apoyo moral y económico.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

El movimiento scout surgió en Inglaterra en el año 1907, gracias a Robert Stephenson Smith Baden-Powell, un general británico que estaba preocupado por la delincuencia y poca formación moral de la juventud de la época, que mayoritariamente trabajaba largas jornadas de tiempo en fábricas y lugares similares. Su objetivo principal era desarrollar un tipo de formación que, a grandes rasgos, creara buenos ciudadanos. Esto fue reflejado en el libro que publicó en el año 1908, titulado *Escultismo para muchachos*. Unos meses antes, en 1907, se celebró el primer campamento scout de la historia, en la isla de Brownsea, donde Baden-Powell llevó a un grupo de 22 jóvenes de entre 11 y 15 años, para poner en práctica sus ideas. [1]

Hoy en día, el movimiento scout está extendido por todo el mundo, constituido por una gran variedad de grupos (se calcula que hay más de 36,5 millones de scouts en el mundo [2]) con sus respectivas tradiciones y creencias específicas, pero manteniendo siempre la esencia establecida por Baden-Powell en el pasado siglo.

Este TFG está enfocado en la Asociación Grupo Scout Estrella Polar, a la que el alumno Miguel Herrera Carrillo, autor del proyecto, pertenece desde el año 2007 y en la que ejerce el cargo de presidente durante este curso 2019/2020.

Figura 1: Logo de la Asociación Grupo Scout Estrella Polar

El grupo scout Estrella Polar fue fundado en el año 1973, y se localiza en el distrito de Ciudad Lineal de Madrid. El equipo de trabajo que mantiene el grupo año tras año está compuesto por jóvenes voluntarios mayores de edad, que planifican y desarrollan actividades y acampadas los fines de semana a lo largo de todo el curso escolar.

El grupo está dividido en **unidades** según la edad de los participantes, estando así compuesto por:

- **Castores**: de los 4 a los 7 años
- **Manada** o **Lobatos**: de los 7 a los 11 años
- **Tropa**: de los 11 a los 15 años
- **Pioneros** o **Red**: de los 15 a los 18 años
- **Clan**: de los 18 a los 19 años
- **Kraal:** equipo de trabajo del grupo (monitores)

Dentro del **Kraal**, aparte de la asignación de una unidad de la que se es monitor, están establecidos unos **cargos**, que separan las distintas tareas que hay que desarrollar a lo largo del año. Los principales son: **jefatura y subjefatura de grupo**, **tesorería**, **secretaría**, **intendencia**, **tiendas de campaña**, **botiquín** y **uniformidad**.

La motivación principal del proyecto surge de la necesidad de tener una plataforma online en la que se puedan unificar y facilitar las tareas de uso común llevadas a cabo en el grupo durante el año, así como tener un portal de usuarios que albergue al conjunto de miembros de la asociación y sus familiares.

Dicha plataforma ha sido llamada **Phoenix**, en referencia al nombre que tiene la unidad de **Kraal** en el grupo, denominada Kraal Fénix. Se puede acceder a través de la página web oficial del grupo scout Estrella Polar, en el siguiente enlace: https://gsestrellapolar.es/phoenix

Figura 2: Logo de la plataforma Phoenix

Si el tribunal así lo requiere, puede acceder a la página registrándose como invitado con una cuenta de Google, seleccionando en la página de registro que es de tipo de usuario FAMILIAR, y en el apartado "scouts de los que eres familiar", escribiendo el nombre Miguel Herrera y la unidad Kraal. Si además se desean probar las funcionalidades de administrador, pueden solicitarlo enviando un correo al email del alumno: miguel.herrerac@estudiante.uam.es, que les aceptará una cuenta temporal como KRAAL para que se puedan observar el resto de características de la aplicación. También, si así se desea, se les puede proporcionar el enlace del repositorio en GitHub donde se encuentra almacenado el código.

1.2 Objetivos

El objetivo principal que se pretende conseguir con la aplicación es el siguiente: unificar, simplificar, y facilitar las tareas llevadas a cabo por la asociación, de una manera intuitiva, segura y accesible.

Más en detalle, los objetivos se pueden dividir en los siguientes aspectos:

- Accesibilidad: la aplicación estará dirigida a tres tipos diferentes de usuarios: los niños y niñas inscritos en el grupo, sus familiares, y el equipo de monitores (Kraal). Por lo tanto, se deben establecer distintas funcionalidades para cada tipo de usuario. Además, se debe tener en cuenta que se pueda acceder a la aplicación de manera sencilla, por lo que el diseño debe de ser intuitivo y la ejecución debe ser viable en la mayor cantidad posible de dispositivos del panorama tecnológico actual.

- Utilidad: las funcionalidades que se implementen deben proporcionar una ventaja considerable a la manera actual en la que se realizan las distintas tareas de la asociación. Dichos beneficios deben aportar tanto simplicidad como seguridad y mantenimiento de los datos.

- Unificación: la plataforma debe ser un lugar donde se recopila la mayor cantidad de información diversa del grupo scout, elementos tales como documentos, archivos multimedia, eventos, inventario, etc.

- Innovación: la aplicación debe ser desarrollada considerando el marco tecnológico actual, escogiendo aspectos innovadores que se usan hoy en día y teniendo en cuenta una visión de futuro.

1.3 Organización de la memoria

Para la organización de la memoria, se ha seguido una metodología basada en el análisis y diseño de Software, junto con un estado del arte que proporciona el contexto donde se enmarcan todas las decisiones escogidas.

La memoria consta de los siguientes capítulos:

- **Estado del arte:** Sección que describe el marco tecnológico actual en el que decidir el camino para poder llevar a cabo los objetivos del proyecto. Este capítulo se divide en tres apartados: una parte centrada en las aplicaciones multiplataforma, un apartado que desarrolla el ámbito de las aplicaciones web, y, por último, un breve apartado con el contexto de aplicaciones con fines similares a los que se intentan desarrollar.

- **Diseño:** Esta sección consta de dos subsecciones:
  El análisis de los requisitos del proyecto, donde se detallan y especifican los objetivos de la aplicación de una manera más enfocada al desarrollo de esta.
  El diseño del modelo de datos que se va a emplear y la organización de los mismos.

- **Desarrollo:** En este apartado se describen las distintas etapas de la implementación de la aplicación, mostrando las decisiones tomadas y lo acontecido durante su desarrollo.

- **Integración, pruebas y resultados:** Esta parte de la memoria hace hincapié en las diferentes pruebas que se han ido llevando a cabo, así como los resultados de la ejecución con usuarios reales.

- **Conclusiones y trabajo futuro:** Para concluir, en este apartado se detallan las diferentes conclusiones a las que se han llegado tras la realización del proyecto, añadiendo a su vez los aspectos que han quedado sin finalizar y las ideas que se desean implementar en versiones futuras.

- **Anexos:** Los anexos incluyen información adicional que complementa el contenido expuesto en la memoria.

2 Estado del arte

2.1 Software multiplataforma

La creciente popularidad en el uso del smartphone, o teléfono inteligente, es sin duda un aspecto clave a tener en cuenta en el desarrollo de software. Por lo tanto, para analizar las distintas tecnologías existentes a día de hoy en el ámbito de las aplicaciones informáticas, se tiene que abordar el denominado tema del software multiplataforma.

Desde que se popularizó el uso del teléfono inteligente durante la pasada década, caben destacar tres grandes alternativas en cuanto al desarrollo de aplicaciones que admitan múltiples plataformas:

- **Aplicaciones nativas**
  Este es el paradigma clásico en el cual la aplicación está específicamente diseñada y desarrollada para un sistema operativo concreto. Si hablamos de smartphones, los sistemas operativos más comunes son Android de Google, donde se usa el lenguaje de programación Java para sus aplicaciones, IOS de Apple, donde se usa el lenguaje Objective C, y, por último, Windows Phone, con entornos de desarrollo .NET. [4]

Figura 3: Sistemas operativos en smartphones [3]

  La principal ventaja de este tipo de aplicaciones es la capacidad de usar el hardware de los dispositivos de manera óptima, ya que cada aplicación está centrada en el dispositivo concreto y todas las características del mismo que pueda aprovechar.

  El punto negativo es, sin duda, la necesidad de programar la aplicación tantas veces como sistemas operativos diferentes se requieran, por tanto, para aplicaciones relativamente sencillas que buscan poder ser ejecutadas en la mayor cantidad posible de dispositivos, es una opción desfavorable.

- **Aplicaciones web**
  Este tipo de aplicaciones se desarrollan mediante tecnologías web como HTML5 (HyperText Markup Language), CSS3 (Cascading Style Sheets) y JavaScript. Están almacenadas en servidores remotos y se accede a ellas mediante una única URL (Uniform Resource Locator). En la siguiente sección se desarrollará este apartado con más detalle.

- **Aplicaciones híbridas**
  Esta clase de aplicaciones intenta sacar el mayor partido a las aplicaciones web y a las nativas. Si bien son diseñadas mediante técnicas de desarrollo web, usan un contenedor nativo para ser ejecutadas, pudiendo así acceder a las funcionalidades y al hardware específico del dispositivo. [5]

  El entorno de desarrollo más popular para aplicaciones híbridas es Apache Cordova, que consta de un contenedor nativo que alberga el código web, y una API (Application Programing Interface) JavaScript que se encarga de conectar y posibilitar la comunicación entre el código web y la API del sistema nativo.

  Aparte de Apache Cordova, otros ejemplos de entornos de desarrollo para este tipo de aplicaciones incluyen PhoneGap, Appcelerator y Appspresso. [4]

2.2 Aplicaciones web

Las aplicaciones web se diferencian de las llamadas aplicaciones nativas en que, mientras que las nativas están específicamente diseñadas para una plataforma concreta, las aplicaciones web se cargan en un servidor externo y llegan al usuario a través de Internet, ejecutándose en su navegador. Esto hace que tengan multitud de beneficios, tales como la accesibilidad a través de diversas plataformas y navegadores, y el hecho de no ser necesaria su instalación en el dispositivo. [6]

Las aplicaciones web surgieron a principios de los años 90, en la forma de páginas HTML estáticas que mostraban texto, añadiendo más adelante la posibilidad de mostrar imágenes y multimedia. Poco tiempo después, en 1995, se creó el lenguaje de programación JavaScript, un lenguaje interpretado y ejecutado en el lado del cliente, que habilitó el uso de elementos dinámicos en la web [7]. A partir de 2005, con la creación de Ajax (Asynchronous JavaScript And XML), se introdujo el concepto de aplicaciones asíncronas, ya que con esta tecnología se permitía actualizar elementos de la página sin necesidad de recargarla en su totalidad [8]. Finalmente, en el año 2015, Alex Russell y Frances Berrman acuñaron el término “aplicaciones web progresivas”, o “Progressive Web Applications” (PWA), para describir las aplicaciones que sacaban partido a las nuevas funcionalidades que tenían los navegadores, difuminando así las diferencias apreciadas por el usuario entre las aplicaciones web y las nativas. [9]

2.2.1 Aplicaciones web progresivas (PWA)

Como se ha explicado antes, el concepto de “aplicación web progresiva”, fue establecido por Alex Russel (ingeniero de Google Chrome) y el diseñador Frances Berrman en 2015, siendo su objetivo principal la creación de aplicaciones web que se comporten y se parezcan a las aplicaciones instaladas en los dispositivos. [10]

La principal característica de este tipo de aplicaciones es la posibilidad de, tras acceder a la URL de la aplicación por primera vez, añadirla a la pantalla de inicio del dispositivo, y así poder ejecutarla como una app nativa. Esto supone una gran diferencia con respecto a las aplicaciones web convencionales, puesto que, mientras que para acceder a una página web clásica debes entrar a través de su URL y descargar el contenido de esta cada vez que se visita, para usar una PWA solo basta con acceder una primera vez y, tras añadirla a la

pantalla de inicio, se descarga en el dispositivo todo el contenido de la aplicación, tanto el estático (ficheros HTML, CSS, JavaScript, imágenes, etc), como el dinámico, que se almacena mediante memoria caché. Por lo tanto, esto hace que la aplicación se pueda ejecutar offline, eso sí, sin actualizar el contenido dinámico hasta que no se disponga nuevamente de conexión.

Desde un punto de vista técnico, las PWA están compuestas, como mínimo, por un manifiesto JSON (JavaScript Object Notation) con metadatos, y con el denominado "service worker", un archivo JavaScript que se encarga de manejar todo lo necesario para el funcionamiento offline de la aplicación. [11]

Mientras que el rendimiento es levemente menor al de las aplicaciones nativas (dependiendo de una conexión estable a la red), las PWA son considerablemente menos pesadas, y, junto con las posibilidades de añadir funcionalidades tales como el uso de notificaciones push o el GPS del dispositivo para obtener ubicaciones, hacen de este tipo de aplicaciones una opción muy prometedora de cara al futuro.

2.2.2 Single Page Applications (SPA)

Las aplicaciones de página única, o single page applications (SPA), se caracterizan por cargar todos los recursos de la aplicación web en la consulta inicial, como si se tratara sólo de una página. Los componentes cambian dinámicamente a medida que el usuario va interaccionando con ellos.

Este paradigma evita las interrupciones causadas por la carga en el servidor de cada página por separado que la aplicación tenga, ya que simula el contenido de las diferentes rutas, posibilitando al usuario navegar hacia delante o atrás de manera casi inmediata, proporcionando de esta manera una experiencia de usuario mucho más fluida. [12]

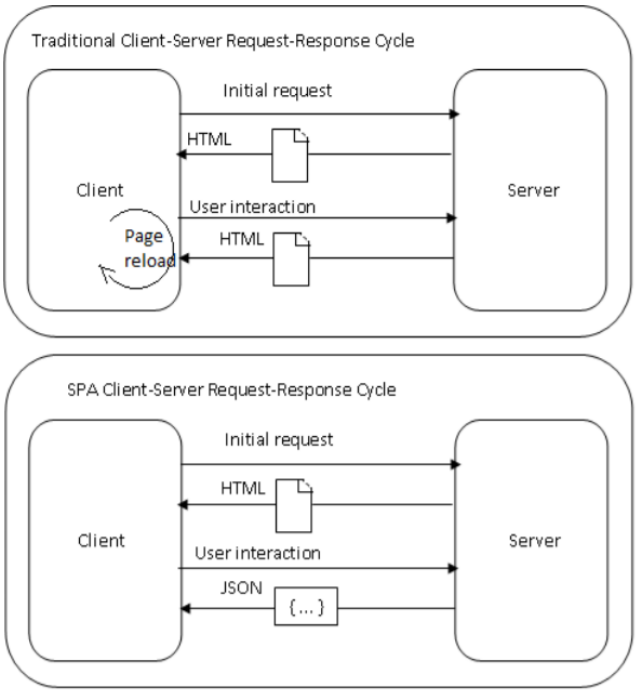

Figura 4: Ciclo de Petición-Respuesta Cliente-Servidor [12]

2.2.3 Frontend en aplicaciones web

Tomando una perspectiva enfocada en el lado del cliente, es decir, en el frontend, hay una gran cantidad de alternativas disponibles para desarrollo de aplicaciones web.

De toda la multitud de posibilidades, hoy en día podemos resaltar tres conjuntos de bibliotecas y herramientas, también llamados *frameworks*, que tienen bastante popularidad y posibilitan la inclusión de todas las tecnologías expuestas previamente, tales como las PWA y las SPA.

- **Vue.js**
  Vue.js se define como un entorno de trabajo progresivo para el desarrollo de interfaces de usuario [13]. Se trata de uno de los frameworks de JavaScript más recientes, que, sin embargo, no deja de ganar popularidad. Esto se debe a que tiene una estructura relativamente simple, contando con una documentación muy clara y detallada, lo cual lo hace muy atractivo para programadores que no cuentan con demasiada experiencia en el desarrollo web. Cuenta además con la librería gráfica Vuetify, con multitud de opciones para personalizar la interfaz.

Figura 5: Logo de Vue.js [13]

- **React**
  React se define como una librería JavaScript diseñada para la construcción de interfaces de usuario [14]. Lanzado por Facebook, y usado a su vez en aplicaciones tales como Instagram, React cuenta con una considerable popularidad entre la comunidad de desarrolladores web.

  Destaca por ser el framework que introdujo una arquitectura centrada en componentes, característica que ha sido adquirida por multitud de otros entornos de trabajo.

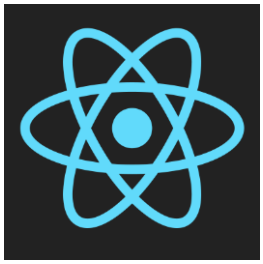

Figura 6: Logo de React [14]

- **Angular**
  Precedido por AngularJS en 2009, Angular fue reescrito en 2016, lanzado por Google. Se define como un framework de diseño de aplicaciones y una plataforma de desarrollo para crear "single page applications" eficientes y sofisticadas. [15]

  Cabe destacar que este entorno es empleado por empresas como PayPal, Microsoft, y, por supuesto, Google.

Figura 7: Logo de Angular [15]

2.2.4 Backend en aplicaciones web

Para hablar sobre el lado del servidor y las diferentes opciones disponibles hoy en día que se adapten a las tecnologías previamente expuestas, se han elegido dos alternativas que se desarrollarán a continuación:

- **Backend as a service (BaaS)**
  Este tipo de enfoque está diseñado especialmente para software multiplataforma que requiera usar un backend alojado en la nube y con funcionalidades recurrentes en las apps de hoy en día, tales como el uso de notificaciones push o la gestión de usuarios. La principal ventaja de esta tecnología es el hecho de que el desarrollador no necesita programar ningún backend, sino que, al suscribirse al servicio, la plataforma ya otorga al usuario multitud de posibilidades para alojar los datos de su aplicación de forma segura, además de otras opciones como mostrar analíticas, autenticaciones de usuarios, hosting web, etc. [16]

  Las dos plataformas más usadas los últimos años que incluyen estos servicios son Amazon Web Services (AWS) y Firebase de Google.

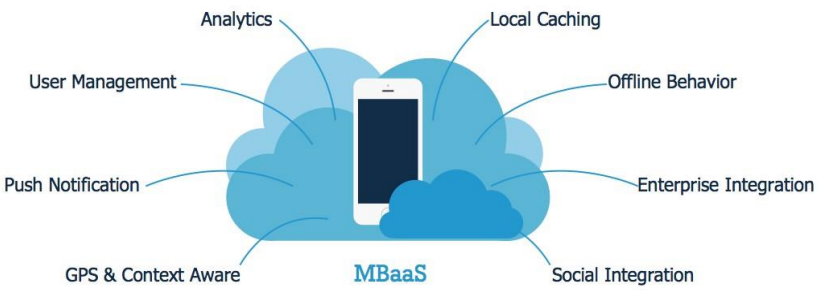

Figura 8: Mobile Backend as a Service [17]

- **Node.js**
  Node.js es definido como un entorno de ejecución para JavaScript construido con el motor JavaScript V8 de Chrome [18]. Una de sus principales características es la utilización de JavaScript en el lado del servidor, a diferencia de otros paradigmas que usan la combinación de lenguajes como PHP y SQL, u otros como Python o Java para el manejo del backend. Resalta por ser una tecnología liviana y eficiente, orientada a eventos, permitiendo elevados niveles de concurrencia, ya que está diseñado para manejar adecuadamente sistemas asíncronos.

  Adicionalmente, cuenta con npm (Node Package Manager), un gestor de paquetes JavaScript que contiene, a día de hoy, más de un millón de paquetes, haciéndolo así el registro de software más amplio del mundo. [19]

  Para desarrollar un backend con Node.js, el entorno más popular es Express.js, que simplifica el desarrollo de servicios web y el intercambio de información, y que, gracias a Node.js, aporta ventajas como la unificación de lenguajes en el frontend y en el backend, ya que ambos usan JavaScript, y su diseño minimalista.

Figura 9: Logo de Node.js [18]

2.3 Aplicaciones similares

En cuanto a la cuestión de sistemas que proporcionen servicios similares a los objetivos del proyecto, cabe destacar la opción de poder utilizar los servicios que proporciona Google, tales como Google Drive, que podría usarse como almacenaje de archivos importantes, Google Calendar, para implementar un calendario del grupo scout, Google Docs y similares, para poder crear documentos de diversos tipos, entre otras opciones.

Dichas aplicaciones y herramientas externas aportan un gran inconveniente en cuanto a uno de los objetivos del proyecto, la unificación. Es preferible tener una única plataforma que albergue el conjunto de dichas funcionalidades, para no tener que depender de sistemas externos.

De todos modos, el uso de un “Backend as a Service”, como es el caso de Firebase, sí que sería recomendable, ya que integraría a la perfección todos los datos almacenados necesarios con la aplicación en sí.

3 Diseño

Tras haber analizado el conjunto de tecnologías existentes hoy en día que guardan relación con los objetivos del proyecto, se van a desarrollar las etapas del análisis y el diseño de la aplicación, y, para concluir, se va a justificar la elección de las tecnologías que finalmente se han escogido para el desarrollo de la plataforma.

3.1 Análisis: Especificación de Requisitos del Software

3.1.1 Descripción general

3.1.1.1 Propósito y ámbito del sistema

La aplicación web Phoenix tiene como propósito principal el crear una red de usuarios que contenga el conjunto de miembros del grupo scout Estrella Polar, y en la que se puedan llevar a cabo multitud de funcionalidades.

Dichas funcionalidades se especificarán más adelante en el apartado de requisitos funcionales, pero, a grandes rasgos, el sistema deberá contar con:

- **Sistema de registro:** Sólo se permitirá el acceso al contenido de la aplicación a usuarios registrados en la misma, ya que la plataforma está destinada sólo a los miembros del grupo scout, incluyendo a los familiares.

- **Administración por el kraal:** Todo el grupo de monitores tendrá el mismo poder para administrar la aplicación, pudiendo editar todo el contenido dinámico de la misma.

- **Validación de miembros de kraal:** Para registrarse como miembro de Kraal en la aplicación, es decir, como administrador, esa solicitud de registro deberá ser aceptada por otro miembro de kraal ya registrado y aceptado.

3.1.1.2 Definiciones, acrónimos y abreviaturas

En esta sección se enumeran el conjunto de palabras clave recurrentes en la aplicación, junto con su significado.

- **Tipo de usuario / Rol:** Podrán ser de tres tipos:

    - **Kraal:** Los administradores de la aplicación, estará compuesto por los monitores y monitoras activos del grupo. Tendrán la capacidad de editar el contenido de las diferentes secciones de la plataforma.

    - **Scout:** Los miembros del grupo scout, tendrán acceso a la edición de su perfil y a visualizar las diferentes secciones de la plataforma, excepto la sección de cargos.

    - **Familiar:** Los familiares de los miembros del grupo scout, deberán indicar los nombres y las unidades de los miembros de los que son

familiares. Tendrán acceso a la edición de su perfil y a visualizar las diferentes secciones de la plataforma, excepto la sección de cargos.

- **Unidad:** Conjunto dentro del grupo scout delimitado según la edad de sus miembros. Cada usuario deberá indicar o bien su unidad (rol Scout), o bien la unidad del scout del que es familiar (rol Familiar), o bien la unidad de la que es monitor (rol Kraal).

    - **Castores / Colonia:** Niños y niñas de entre 4 y 7 años.

    - **Manada / Lobatos:** Niños y niñas de entre 7 y 11 años.

    - **Tropa:** Jóvenes de entre 11 y 15 años.

    - **Pioneros / Red:** Jóvenes de entre 15 y 18 años.

    - **Clan / Rovers:** Jóvenes de 18 o 19 años que se dedican a aprender el funcionamiento del kraal.

    - **Kraal:** Jóvenes mayores de edad que se dedican a gestionar el grupo. Cada miembro tiene un cargo y es monitor de una unidad. En la aplicación, los miembros del Kraal escogerán como rol "Kraal" y como unidad la unidad de la que son monitores.

    - **Apoyo:** Exmiembros del grupo que no están en la unidad de kraal pero se dedican a ayudar en lo que haga falta ocasionalmente.

- **Cargo:** Cargo asignado a un miembro del Kraal que indica un tipo de tarea de la que este es responsable. La aplicación contará con la implementación de funcionalidades para los cargos de:

    - **Intendencia:** Cargo dedicado a la gestión del material del grupo scout.

    - **Tiendas:** Cargo dedicado a la gestión de las tiendas de campaña del grupo.

    - **Uniformidad:** Cargo dedicado a la gestión de las camisas de uniforme, pañoletas e insignias del grupo.

    - **Botiquín:** Cargo dedicado a la gestión de todo lo relacionado con medicinas y material sanitario del grupo.

- **Promesa:** La promesa scout es la progresión más importante en un grupo scout, se representa mediante una pañoleta con los colores del grupo, y se puede conseguir en un campamento a partir de la unidad de tropa, eligiendo dos padrinos que ya tengan la promesa y yendo a una velada especial a hablar sobre tu compromiso con el grupo. En la aplicación se podrá indicar si la tienes o no, y la fecha en la que la obtuviste.

- **Padrinos / Madrinas:** Los padrinos o madrinas son dos scouts con promesa que el scout que se dispone a pedir la misma elige para que actúen como sus mentores. En la aplicación, si los padrinos de un usuario están registrados, podrán ser elegidos para que aparezcan en su perfil, si no, se indicará únicamente sus nombres.

- **Evento:** Evento que se podrá añadir al calendario de grupo, pudiendo durar entre unas horas o varios días, y estar destinado a una unidad concreta o a todo el grupo.

- **Encuesta:** Encuesta dirigida a usuarios de una o varias unidades, que constará de una pregunta y varias opciones posibles entre las que se deberá elegir una para poder votar.

- **Acampadas:** Sección con una lista de eventos de más de un día de duración realizados por el grupo scout.

- **Rutas:** Sección con una lista de diferentes rutas por el campo que incluirán información como el lugar, la duración y la dificultad de esta.

- **Objetos perdidos:** Sección de la aplicación con una lista de objetos extraviados por algún scout en algún evento pasado, que estarán acompañados por una foto. El usuario que lo haya perdido podrá reclamarlo.

- **Música:** Sección de música con las letras y acordes de canciones populares en el grupo scout.

- **Galería:** Sección con vídeos de YouTube relacionados con algo del grupo scout.

3.1.2 Requisitos funcionales

Para analizar los requisitos funcionales de la aplicación, se han enumerado detalladamente todas las distintas funcionalidades que se podrán llevar a cabo en la misma, dividiéndolas en tres partes según el usuario que las pueda realizar: primeramente, los usuarios no registrados, a continuación, todo el conjunto de usuarios registrados y, por último, los usuarios de tipo Kraal, es decir, los administradores.

El conjunto de dichos requisitos ha sido elaborado teniendo en cuenta las aportaciones y sugerencias de distintos miembros del grupo scout. Tanto al kraal como a miembros del grupo se les envió un correo electrónico con un formulario donde había que responder con propuestas de qué es lo que les gustaría que se pudiera hacer en la aplicación.

A continuación, se enumeran los requisitos funcionales desarrollados:

Para usuarios no registrados

1. Crear una cuenta en la aplicación
2. Iniciar sesión en la aplicación

Para todos los usuarios registrados

1. Cerrar sesión en la aplicación
2. Visualizar la página de perfil propia
3. Poder entrar desde la página de perfil de un usuario a la página de perfil de sus padrinos si están registrados
4. Ver la foto de perfil de un usuario en mayor resolución, desde la página de perfil de este
5. Editar la información del perfil propio
6. Subir una foto de perfil desde el dispositivo
7. Elegir una foto de perfil de imágenes de perfil estáticas disponibles en la plataforma
8. Restablecer la foto de perfil de Google
9. Subir una foto de portada desde el dispositivo
10. Editar la información del perfil (nombre, unidad, cargo, teléfono, estado, año de entrada en el grupo, fecha de obtención de la promesa)
11. Elegir como padrino a un usuario registrado en la aplicación
12. Eliminar la cuenta propia de manera permanente
13. Cambiar el tema de la aplicación a modo oscuro o claro
14. Visualizar las páginas de las distintas unidades con su información
15. Visualizar en las páginas de las unidades sus miembros, pudiendo entrar a su página de perfil
16. Visualizar el calendario con los eventos que correspondan según la unidad del usuario
17. Cambiar la visualización del calendario a día, 4 días, semana, o mes
18. Visualizar las encuestas disponibles dirigidas a la unidad del usuario
19. Poder votar las encuestas disponibles dirigidas a la unidad del usuario que no hayan sido votadas ya por este
20. Visualizar los resultados de las encuestas dirigidas a la unidad del usuario que ya ha votado
21. Visualizar acampadas
22. Entrar en la página de una acampada
23. Ver el dossier de una acampada
24. Visualizar rutas
25. Visualizar objetos perdidos
26. Reclamar/Dejar de reclamar un objeto perdido
27. Entrar en la página de perfil del usuario que ha reclamado el objeto perdido
28. Visualizar canciones
29. Ver la letra de una canción
30. Ver la letra con los acordes de una canción
31. Visualizar galería
32. Reproducir el vídeo de la galería en YouTube

Para todos los usuarios aceptados de tipo Kraal

1. Visualizar solicitudes para entrar como Kraal
2. Aceptar/Rechazar solicitudes para entrar como Kraal
3. Visualizar inventario de intendencia por categoría
4. Visualizar lista de la compra de intendencia
5. Añadir objeto de intendencia
6. Editar objeto de intendencia

7. Eliminar objeto de intendencia
8. Generar lista de objetos de intendencia en pdf
9. Generar lista de objetos de intendencia en hoja de cálculo
10. Visualizar inventario de uniformidad por categoría
11. Visualizar lista de la compra de uniformidad
12. Añadir objeto de uniformidad
13. Editar objeto de uniformidad
14. Eliminar objeto de uniformidad
15. Generar lista de objetos de uniformidad en pdf
16. Generar lista de objetos de uniformidad en hoja de cálculo
17. Visualizar inventario de botiquín por categoría
18. Visualizar lista de la compra de botiquín
19. Añadir objeto de botiquín
20. Editar objeto de botiquín
21. Eliminar objeto de botiquín
22. Generar lista de objetos de botiquín en pdf
23. Generar lista de objetos de botiquín en hoja de cálculo
24. Visualizar inventario de tiendas de campaña por unidades
25. Añadir tienda de campaña
26. Editar tienda de campaña
27. Eliminar tienda de campaña
28. Ver la hoja de revisión de una tienda de campaña
29. Ver tiendas de campaña para llevar a arreglar
30. Ver tiendas de campaña para arreglar por el grupo scout
31. Generar lista de tiendas en pdf
32. Generar lista de tiendas en hoja de cálculo
33. Visualizar todos los eventos del calendario
34. Añadir eventos en el calendario
35. Editar eventos en el calendario
36. Eliminar eventos en el calendario
37. Visualizar todas las encuestas existentes
38. Añadir encuestas
39. Visualizar participantes de las encuestas
40. Finalizar encuestas para que no puedan ser votadas
41. Eliminar encuestas
42. Añadir acampadas
43. Editar acampadas
44. Eliminar acampadas
45. Ver documentos de las acampadas por categoría
46. Subir documentos de las acampadas a su categoría
47. Eliminar documentos de las acampadas
48. Añadir rutas
49. Editar rutas
50. Eliminar rutas
51. Añadir objetos perdidos
52. Editar objetos perdidos
53. Eliminar la reclamación de un usuario a un objeto perdido
54. Eliminar objetos perdidos
55. Añadir canciones
56. Editar canciones
57. Eliminar canciones

58. Añadir videos en la galería
59. Eliminar videos de la galería

Adicionalmente, en el Anexo A, se muestra el diagrama de Casos de Uso del sistema.

3.1.3 Requisitos no funcionales

Los requisitos no funcionales están compuestos por todas las propiedades que el sistema debe incluir para que los requisitos funcionales puedan llevarse a cabo sin problemas.

- Requisitos de interfaz y usabilidad:
  - Tener una interfaz intuitiva y fácil de usar sin necesidad de explicaciones detalladas.
  - Contar con una barra de navegación superior y lateral, siguiendo el patrón de los diseños comunes en las aplicaciones actuales.
  - Estar ilustrada con imágenes y fotos del grupo en las diferentes secciones.
  - Adaptar el contenido adecuadamente al tamaño de pantalla pertinente, ya sea ordenador o dispositivo móvil.
  - Usar una paleta de colores consistente.
- Requisitos operacionales:
  - Ser una aplicación multiplataforma, funcional tanto en sistemas operativos distintos como en dispositivos de multitud de tamaños.
- Requisitos de documentación:
  - La aplicación estará en su totalidad en castellano.
- Requisitos de seguridad:
  - Tener en cuenta la información visible de los distintos usuarios.
  - Limitar el uso de contraseñas, usando métodos como el inicio de sesión mediante cuenta de Google.
- Requisitos de mantenibilidad y portabilidad:
  - Tener en cuenta el uso de tecnologías persistentes y una visión de futuro para implementar actualizaciones sin problema alguno.
- Requisitos de recursos:
  - Ser una aplicación ligera en cuanto al tamaño necesario de almacenamiento.

3.2 Diseño del Software

Tras el apartado de análisis, en el que se ha establecido **qué** es lo que se quiere que incluya la aplicación, se va a desarrollar el apartado del diseño, en el que se explica **cómo** se quieren llevar a cabo los requisitos.

3.2.1 Arquitectura del sistema

Para empezar la sección de diseño, se va a exponer la arquitectura del sistema que se va a implementar, destacando dos aspectos relevantes, el patrón de diseño Modelo Vista Vista-Modelo (MVVM) y la denominada arquitectura backendless.

3.2.1.1 Arquitectura MVVM

Para poder definir la arquitectura de un sistema de software es muy importante tener en cuenta un patrón de diseño o modelo de abstracción que nos sirva para poder estructurar de

manera eficaz todos los componentes de la aplicación, tanto la parte de la interfaz (Vista), como la de la lógica (Controlador) y la del modelo de datos (Modelo).

El patrón MVVM se diferencia del clásico MVC (Modelo Vista Controlador), en que, en el MVC, es el Controlador (el componente que se encarga de toda la lógica de la aplicación que enlaza la vista con el modelo) quien debe actualizar manualmente los cambios producidos en el Modelo o en la Vista de manera independiente, mientras que en el patrón MVVM, estos cambios se actualizan automáticamente gracias al componente Vista-Modelo. Es decir, si se produce un cambio en la Vista, los datos son actualizados de forma automática en la parte del Modelo, y viceversa.

Por ejemplo, el entorno de desarrollo Vue.js está centrado en el componente Vista-Modelo del patrón MVVM, que se encarga del trabajo que realiza el Controlador en el diseño MVC, pero de una manera que enlaza los datos entre la Vista y el Modelo de forma reactiva, simple y flexible. [13]

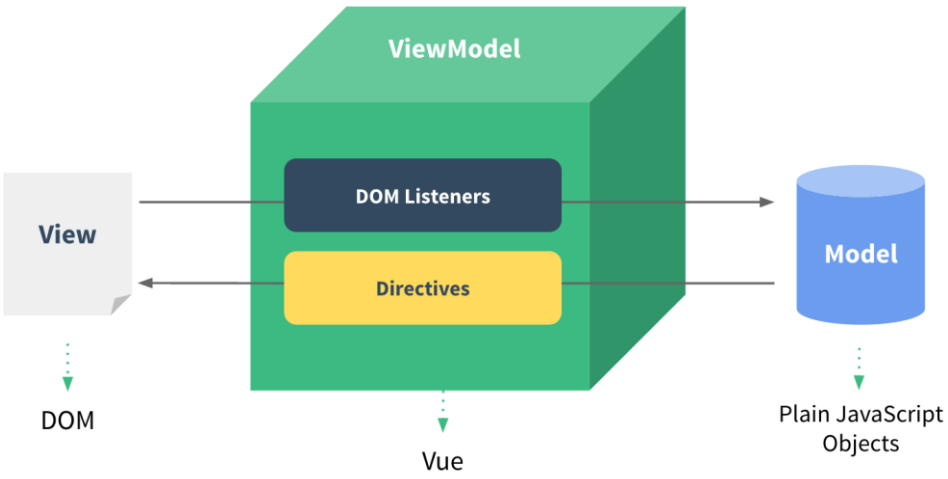

Figura 10: Arquitectura MVVM en Vue.js [13]

3.2.1.2 Arquitectura Backendless

De cara a la parte del servidor de la aplicación (Modelo), se ha optado por un diseño backendless, es decir, se ha prescindido de programar o diseñar un backend personalizado. Se ha elegido la opción de escoger un BaaS (Backend as a Service), en concreto la plataforma Firebase de Google, que aporta multitud de ventajas que se desarrollarán en apartados siguientes.

3.2.2 Modelo y base de datos

El modelo de datos que se va a desarrollar en este proyecto cuenta con once elementos que albergan todo el contenido dinámico de la aplicación, que se deberá almacenar en la base de datos.

Las once entidades se han diseñado como objetos JSON, incluyendo siempre un atributo id que actúa como clave del objeto. La base de datos se compone por los siguientes campos: usuarios, acampadas, botiquín, canciones, encuestas, eventos, objetos, perdidos, rutas, uniformes y videos.

Figura 11: Modelo de datos

3.3 Tecnologías utilizadas

Tras la investigación realizada y las necesidades del proyecto, finalmente se ha escogido la opción de desarrollar una página web progresiva, utilizando el entorno de desarrollo Nuxt.js, de Vue.js, con Vuetify y usando Firebase como backend.

3.3.1 PWA, Vue.js y Nuxt.js

Teniendo en cuenta los requisitos funcionales y no funcionales de la aplicación, y lo expuesto en el estado del arte, la opción de optar por una PWA en modo Single Page Application (SPA), es sin duda alguna una buena idea para el proyecto, ya que la plataforma no es una aplicación a gran escala y uno de sus principales objetivos es que sea sencilla de instalar y de usar en múltiples plataformas.

Para la realización de la PWA se ha elegido el framework Vue.js, debido a que suele ser descrito como un entorno relativamente simple y con una curva de aprendizaje poco pronunciada, además de ser usado por una gran parte de la comunidad de desarrolladores de JavaScript.

- **Vue.js:** Como se ha explicado en apartados anteriores, Vue.js se centra en la parte Vista-Modelo (ViewModel) de la arquitectura del sistema, esto proporciona una sincronización entre la interfaz de usuario y el modelo de datos. Dicho componente se inicializa con la instancia del objeto Vue, que enlaza todos los componentes del DOM (Document Object Model, la interfaz de la aplicación), para que, en cuanto los datos cambien, la interfaz se actualice automáticamente. Dichos datos se representan mediante objetos JavaScript, y es gracias a la instancia del objeto Vue lo que hace que sean reactivos, en cuanto su valor cambia, toda la aplicación en conjunto se sincroniza y actualiza. [13]

- **Nuxt.js:** este framework de alto nivel es un entorno progresivo basado en Vue.js, que organiza y facilita el desarrollo de aplicaciones Vue, por medio del empleo de una estructura clara y personalizada. Permite la configuración de un proyecto Single Page Application y la opción de implementar la aplicación como una PWA de manera muy sencilla, por lo que este framework es una opción recomendable para el proyecto. Nuxt cuenta con funcionalidades de Vue.js como son:

    - **Vue Router:** El sistema de enrutamiento de páginas de Vue.js, que en Nuxt está diseñado de una manera más intuitiva, permitiendo el uso de rutas anidadas o modificar los parámetros de las rutas fácilmente.

    - **Vuex:** el gestor del estado de la aplicación, se detallará en el punto 3.3.4.

  Adicionalmente, Nuxt permite el uso de middleware, que permite administrar el acceso de distintos tipos de usuarios a las páginas de manera rápida y eficaz. [20]

Figura 12: Logo de Nuxt.js [20]

3.3.2 Vuetify

Vuetify se define como la librería de componentes número uno para Vue.js. Se trata de una librería dedicada a la interfaz de usuario de la aplicación, usando la especificación Material Design (lenguaje visual basado en la interfaz del sistema operativo Android). [21]

Permite proporcionar a la aplicación con un aspecto actual, intuitivo y atractivo, y, además, cuenta con una documentación muy completa, con abundantes ejemplos ejecutables.

Figura 13: Logo de Vuetify [21]

3.3.3 Firebase

Firebase es una plataforma de Google dedicada a facilitar el desarrollo de aplicaciones web, ofreciendo diferentes tipos de servicios como una base de datos en la nube, un servicio de hosting para alojar aplicaciones web, un sistema de autenticación de usuarios o un almacenaje de archivos en la nube. [22]

En la aplicación de este proyecto se han elegido los siguientes servicios de Firebase:

- **Firebase Auth:** Este sistema permite gestionar las cuentas de todos los usuarios que usan la aplicación. Dispone de un sistema de autenticación que emplea únicamente código en el lado del cliente, siendo asimismo posible personalizar las reglas de los criterios de acceso de los usuarios. Permite a su vez el uso de una gran cantidad de opciones para poder acceder como usuario a la aplicación a desarrollar, ya sea bien usando un correo y contraseña, una cuenta de Google, una cuenta de Facebook, etc.

  Para este proyecto se ha decidido únicamente posibilitar el acceso con una cuenta de Google, ya que todo el conjunto del grupo scout dispone de una, y de esta manera se hace más simple el proceso de registro e inicio de sesión de los usuarios.

- **Cloud Firestore:** Este sistema funciona como la base de datos de la aplicación. Existe otra opción en Firebase llamada Realtime Database, que es una versión anterior, y se diferencia de Cloud Firestore en que la segunda puede realizar consultas más potentes y usa un modelo de datos más intuitivo.

  Este sistema almacena datos como colecciones de documentos, de una manera muy similar a JSON (JavaScript Object Notation), y permite la edición de reglas de seguridad para lectura y escritura y el uso de filtros e índices. [23]

- **Firebase Storage:** Permite el almacenamiento en la nube de archivos de multitud de tipos como imágenes, vídeos o documentos, de una manera rápida y efectiva. En

este proyecto se usará para almacenar los documentos y las imágenes dinámicas de las diferentes secciones de la aplicación.

Para concluir este apartado, cabe destacar que el uso de Firebase es gratuito, siempre y cuando no se sobrepasen los límites establecidos en su plan de pago. El plan escogido para el proyecto es el denominado plan Spark, plan gratuito si no se exceden los 10000 usuarios autenticados al mes, 1 GiB (2^{30} Bytes) de datos almacenados en Cloud Firestore y 5 GB almacenados en Storage. [24]

Figura 14: Logo de Firebase [24]

3.3.4 Vuex

Vuex se encarga de la administración del estado de aplicaciones Vue.js, disponiendo de un almacenamiento centralizado de todos los componentes de la aplicación, asegurándose que los datos solo puedan ser mutados siguiendo determinadas reglas. [25]

A grandes rasgos, este módulo actúa como intermediario entre el backend (Firebase), y los diferentes componentes de Vue, de tal modo que siempre que se desee hacer una operación con Firebase, esta deberá ser llevada a cabo en el llamado *store* de Vuex, por medio de una acción y una mutación, que a continuación actualizarán los datos del estado de la aplicación. Las acciones son los métodos que son llamados desde el componente de Vue específico, mediante la función *dispatch()*, y que realizan la transacción con la base de datos, posteriormente llamando al método *commit()*. Este método representa el concepto de mutación, que actualiza los datos modificados por la acción en el estado (*state*) de Vuex, el "almacén" local de los datos dinámicos de la aplicación.

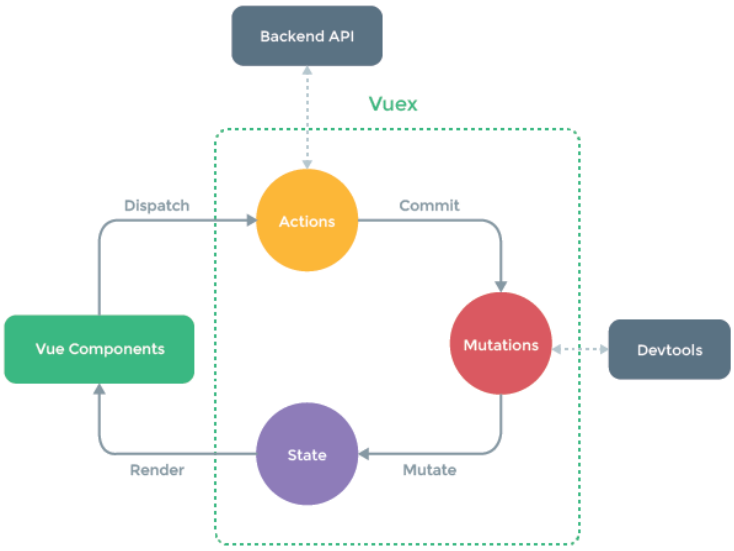

Figura 15: Funcionamiento de Vuex [25]

4 Desarrollo

El desarrollo de la aplicación web Phoenix comenzó el mes de marzo de este año, usando como repositorio la plataforma GitHub. A lo largo del desarrollo del proyecto se han realizado un total de 60 commits, como se puede ver en el siguiente gráfico.

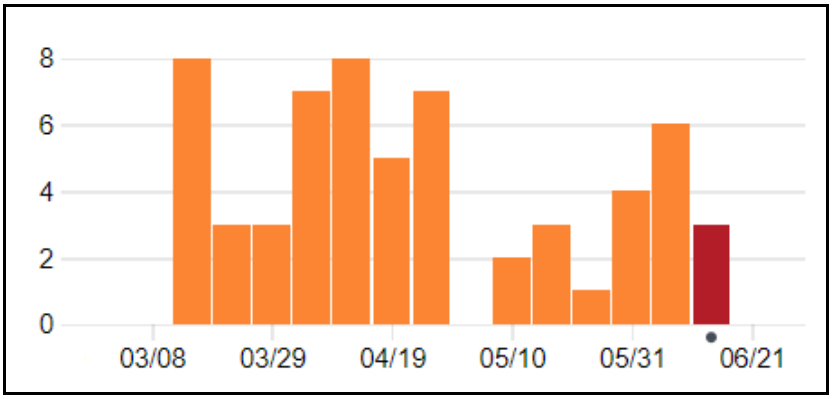

Figura 16: Historial de Commits por semana

La codificación ha sido llevada a cabo mediante distintas fases, cada una centrada en aspectos concretos de la aplicación. Se ha decidido usar el editor de código Visual Studio Code de Microsoft, un programa que está adecuadamente diseñado para la estructuración y programación de aplicaciones web.

4.1 Primeros pasos

Para iniciar el proyecto desde cero, lo primero que se necesitó hacer fue instalar Node.js, puesto que para crear un nuevo proyecto en Nuxt.js, se requiere el uso del gestor de paquetes JavaScript npm, de Node.js. Una vez instalado, para crear la aplicación se ejecutó en la terminal el comando:

```
$ npx create-nuxt-app Phoenix
```

A continuación, se seleccionaron en la terminal las distintas opciones de configuración del proyecto, que en este caso fueron el tener soporte para PWAs, el uso de la librería Vuetify, el modo SPA, y el uso del gestor de paquetes npm.

Tras esta configuración el proyecto fue creado, y, mediante el comando "`npm run dev`", se pudo acceder a la primera versión de la aplicación web, en la dirección http://localhost:3000.

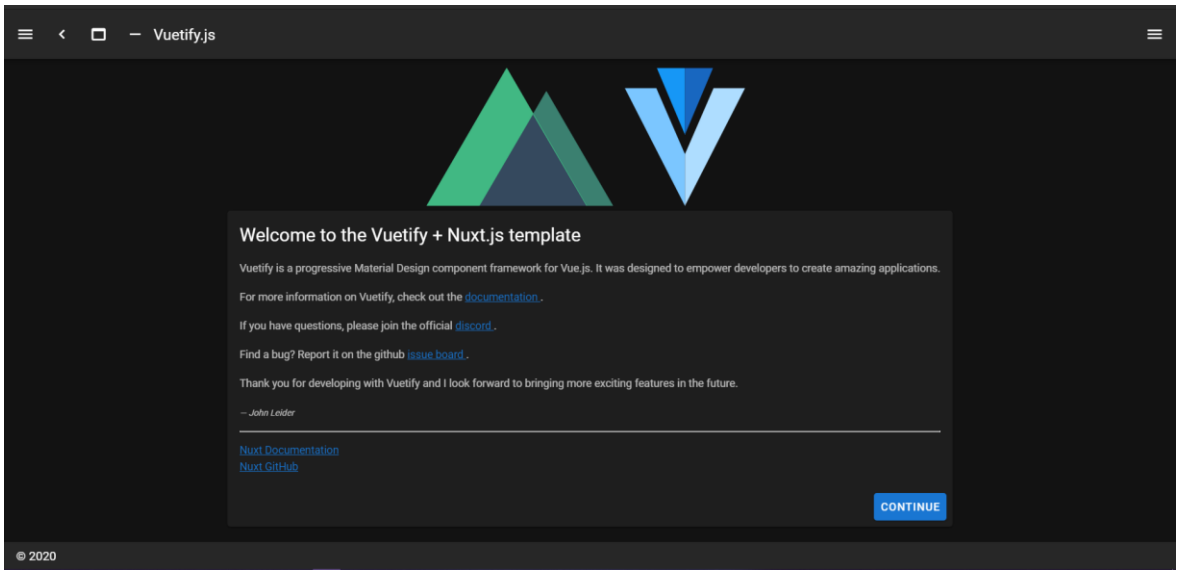

Figura 17: Diseño inicial de aplicación en Nuxt.js con Vuetify

4.2 Gestión de usuarios

La primera etapa de desarrollo del proyecto estuvo dedicada a la administración de usuarios en la aplicación.

4.2.1 Autentificación

Para permitir la autenticación de usuarios en la plataforma, el primer paso fue unir Firebase a la aplicación. Para ello, se tuvo que crear un nuevo proyecto Firebase en su web, mediante una cuenta de Google, y, posteriormente, añadir un fichero de configuración en el proyecto Nuxt, que fue llamado *"firebase.js"*. Este archivo incluye información de las direcciones de los servicios de autenticación, de la base de datos Cloud Firestore y del almacén de ficheros Cloud Storage. Adicionalmente se añadieron los métodos para la inicialización de estos servicios, exportando una variable llamada *"firebase"*, que se podrá emplear dentro de los distintos componentes de la plataforma.

Para posibilitar la autenticación de usuarios por medio de cuentas de Google, se tuvo que habilitar esta opción en el apartado "Sign-in method" de la web de Firebase.

La codificación de este apartado se implementó para que al pulsar el botón "Entrar" o el botón "Iniciar Sesión", de la página de inicio, se llame a una acción de Vuex que inicializa el objeto *"firebase.auth.GoogleAuthProvider()"*, que mediante el método *"firebase.auth().signInWithPopup()"*, abre una ventana en la que el usuario elige una cuenta de Google con la que entrar. Seguidamente, se comprueba si ya existe un usuario con dicho correo en la base de datos. En tal caso, el usuario inicia sesión en su cuenta, y, en caso contrario, se lanza la página *"signup.vue"* de registro para nuevos usuarios.

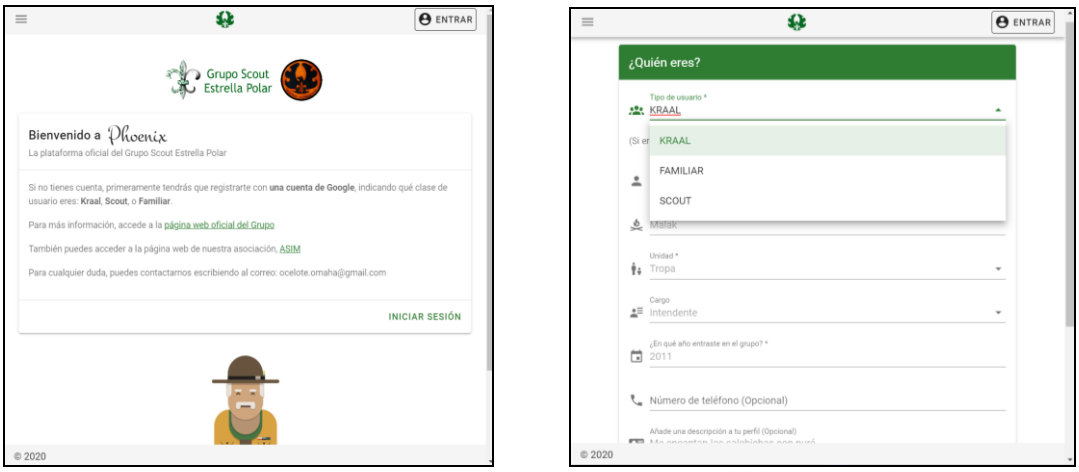

Figura 18: Página de inicio y de registro de Phoenix

4.2.2 Solicitudes de Kraal

Para registrarte como usuario de tipo Kraal, es necesario que alguien ya registrado y aceptado de ese mismo tipo de usuario te acepte. Para ello, cuando un nuevo usuario de rol "kraal" se registra, el campo "aceptado" de ese usuario en la base de datos es "false". Para la aceptación de usuarios, se implementó una página de solicitudes de kraal en la que poder aceptar o rechazar todas las peticiones de usuarios nuevos que desean tener derechos de administrador en la aplicación.

Adicionalmente, se elaboró una burbuja de notificación sobre el avatar de perfil de la barra superior, que indica el número de solicitudes nuevas, mediante el componente de Vuetify "<v-badge>", que, a través de la función *"listaSolicitudesKraal()"*, en el store de Vuex, escucha cuantos usuarios hay en la base de datos con el rol "Kraal" y con el campo *"aceptado"* igual a *"false"*. Finalmente, al aceptar usuarios se cambia el valor de dicho campo a *"true"*, y, al rechazarlos, se les borra automáticamente la cuenta.

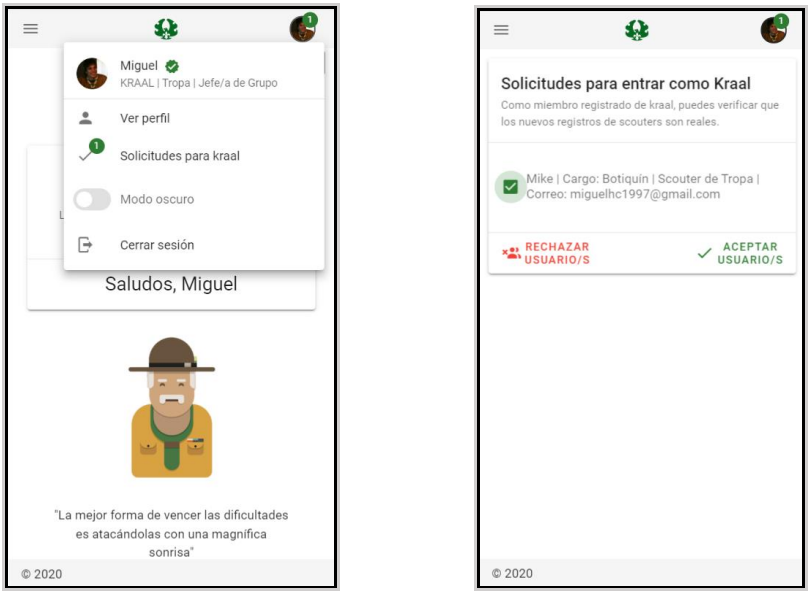

Figura 19: Solicitudes de usuarios de tipo Kraal

4.2.3 Personalización de perfil

Una funcionalidad relevante para poder proporcionar un enfoque de red social a la plataforma es la posibilidad de poder personalizar la información de la cuenta, para que cada usuario pueda dar un estilo personal a su página de perfil.

Para acceder a la ruta de las diferentes páginas de perfil, se añadió el id del usuario correspondiente a la página *"/perfil"*, de tal manera que la dirección de la página de perfil de cierto usuario es la siguiente: *"/perfil/[id del usuario]"*.

En el botón de *"EDITAR PERFIL"*, se accede a la página de edición, donde se puede modificar toda la información de la cuenta, subiendo imágenes de portada o de perfil, o eligiendo como padrinos a usuarios registrados en la plataforma. Las imágenes se suben al Storage de Firebase, en la carpeta *"/perfil"* (para las fotos de perfil), o *"/banners"* (para las imágenes de portada), guardando a su vez en los datos del usuario de la base de datos el nombre del archivo de la imagen y su ruta para poder visualizarla. En caso de cambiar una imagen o borrar la cuenta, todas las fotos subidas en el Storage del usuario se eliminan.

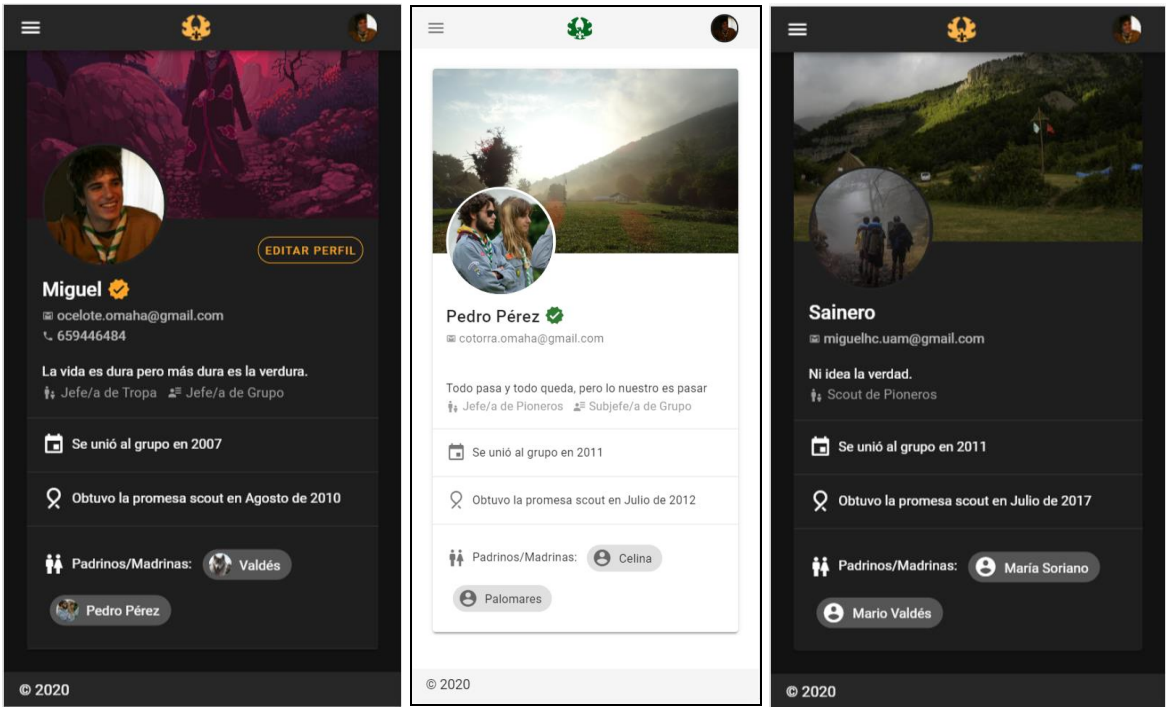

Figura 20: Visualización de varias páginas de perfil

4.3 Unidades

Phoenix consta de diferentes secciones, accesibles desde el panel deslizante de la izquierda. La sección de unidades está diseñada para que se pueda ver la información de cada unidad del grupo y acceder a las páginas de los miembros de dicha unidad, tanto los scouts, como los familiares y los monitores.

Esto se ha implementado por medio del uso del componente "<v-card>" y de listas desplegables *"<v-list>"* y *"<v-list-group>"*

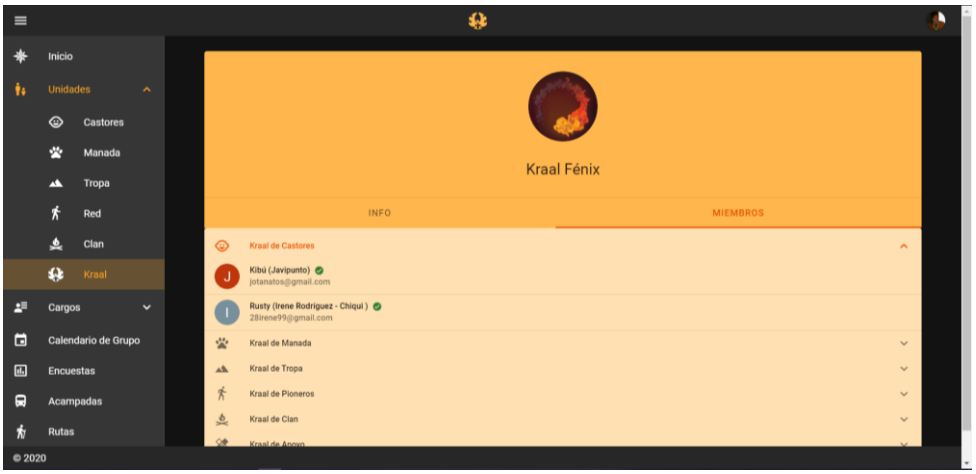

Figura 21: Visualización de miembros de Kraal por unidades

4.4 Cargos

La sección de cargos, solamente accesible por miembros aceptados de Kraal, dispone de apartados distintos para poder realizar inventarios del material del grupo scout.

4.4.1 Intendencia / Uniformidad / Botiquín

Estas tres secciones se han implementado de manera similar, pudiendo añadir el nombre del objeto, su categoría, la cantidad de la que se dispone, y cuantas unidades hacen falta comprarse si es necesario. Dichos objetos se pueden modificar seleccionándolos, y, además, se ha añadido una sección con una lista de la compra, que enumera todos los objetos que se han de comprar y sus cantidades.

Por último, una funcionalidad que se ha añadido mediante el uso de las librerías de JavaScript *"jsPDF"* y *"XLSX"*, es el poder descargar el inventario en formato *".pdf"* o *".xlsx"* (hoja de cálculo), para poder imprimirse y usarse en diferentes ocasiones como en campamentos.

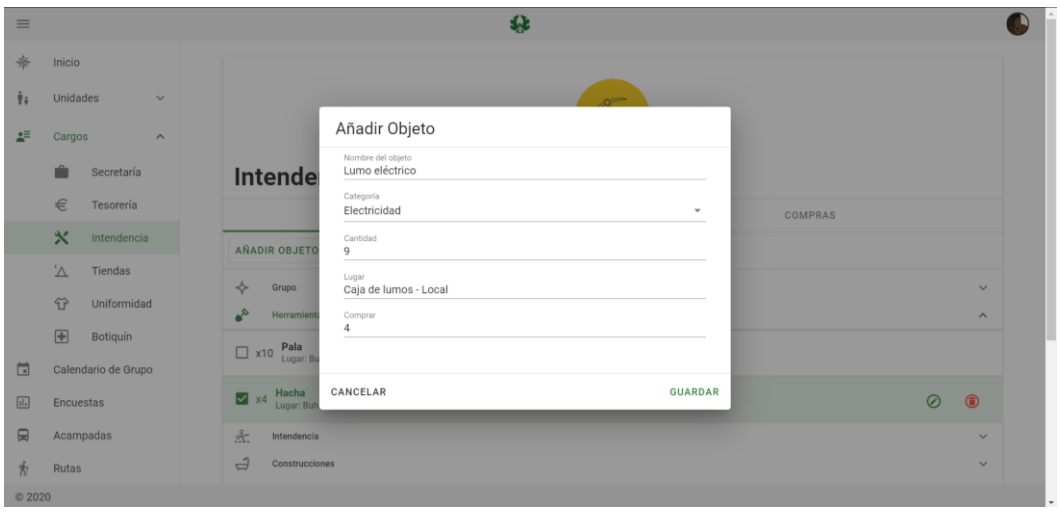

Figura 22: "Añadir objeto" en la página de inventario de Intendencia

4.4.2 Tiendas de campaña

En esta sección, relativamente similar a la anterior, se han añadido las opciones de añadir el estado de cada tienda de campaña, un archivo con su "hoja de revisión", y la posibilidad de indicar si hace falta que se lleve a arreglar a una tienda especializada, o si se puede arreglar por los miembros del grupo. El archivo de cada "hoja de revisión" es subido a la carpeta *"/tiendas"* del Storage de Firebase.

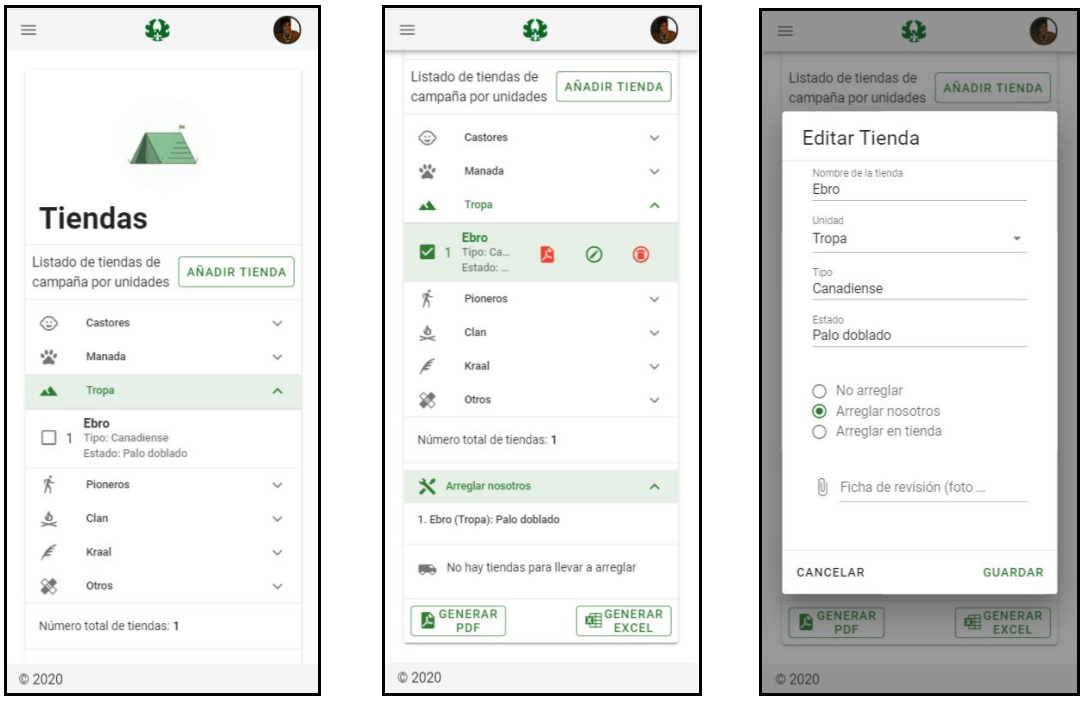

Figura 23: Sección de Tiendas de Campaña

4.5 Calendario

El calendario de grupo es una de las secciones más importantes de la aplicación, pues su función es informar a todos los miembros del grupo y las familias de la fecha de las distintas actividades que se van a realizar o notificaciones importantes. Se ha implementado mediante el componente de Vuetify "VCalendar", que dispone de una gran cantidad de herramientas para poder añadir eventos y personalizarlos.

Para añadir un evento se debe ser un miembro de Kraal, y basta con seleccionar el día requerido en el calendario. A continuación, se podrá poner el nombre del evento con una descripción y las fechas y horas de inicio y finalización (siendo obligatorio únicamente la fecha de inicio, puesto que, si solo se indica esta, se asume que es un evento de día entero). Por último, se debe indicar quien podrá visualizar el evento, cuyas opciones son "el Kraal", "Mi unidad" (los familiares y miembros de la unidad de la que se es monitor), y "Todo el grupo". El evento aparecerá de un color distinto según la unidad a la que esté destinado.

Todos los eventos son guardados en la base de datos, mediante el uso de acciones de Vuex, y son leídos de la misma cuando se accede a la página por primera vez. Si se actualiza algún evento, se actualiza mediante una acción que llama a la base de datos, y se vuelven a cargar todos los eventos desde el *state* del *store* de Vuex.

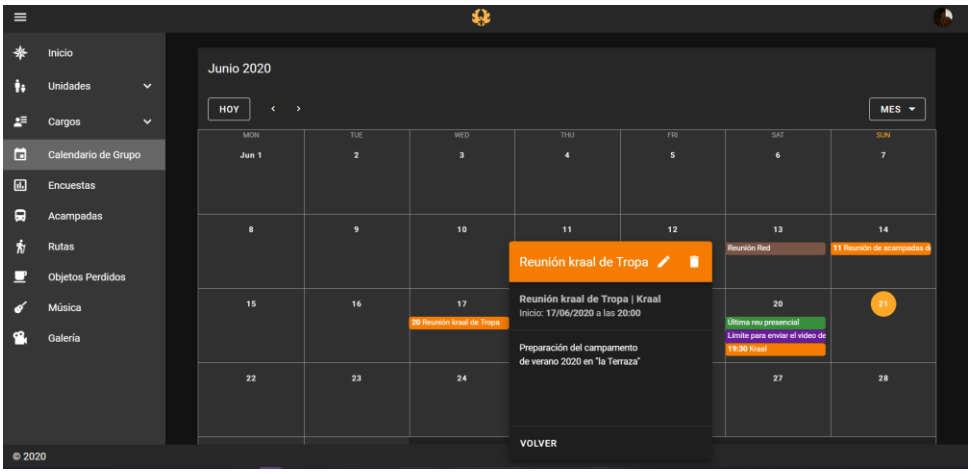

Figura 24: Calendario de grupo

4.6 Encuestas

La sección de encuestas se ha desarrollado por medio de un componente externo llamado Vue-poll, elaborado por Prokopis Pietris, y sacado de la colección de componentes y proyectos de Vue.js, "Made with Vuejs". [26]

Los usuarios de Kraal pueden añadir diferentes encuestas, personalizando el número de opciones y las unidades de los usuarios que pueden votarlas. Una vez publicada, cada usuario al que esté destinada la encuesta podrá votar una opción, que hace que se sume uno a la opción correspondiente del campo "votes" de la encuesta en la base de datos. También se añade el id del usuario que ha votado dicha opción, para poder visualizar qué usuarios han votado cada respuesta. Dichos resultados solo pueden visualizarse por miembros del kraal, que también pueden finalizar las encuestas para que no se pueda votar más, o eliminarlas.

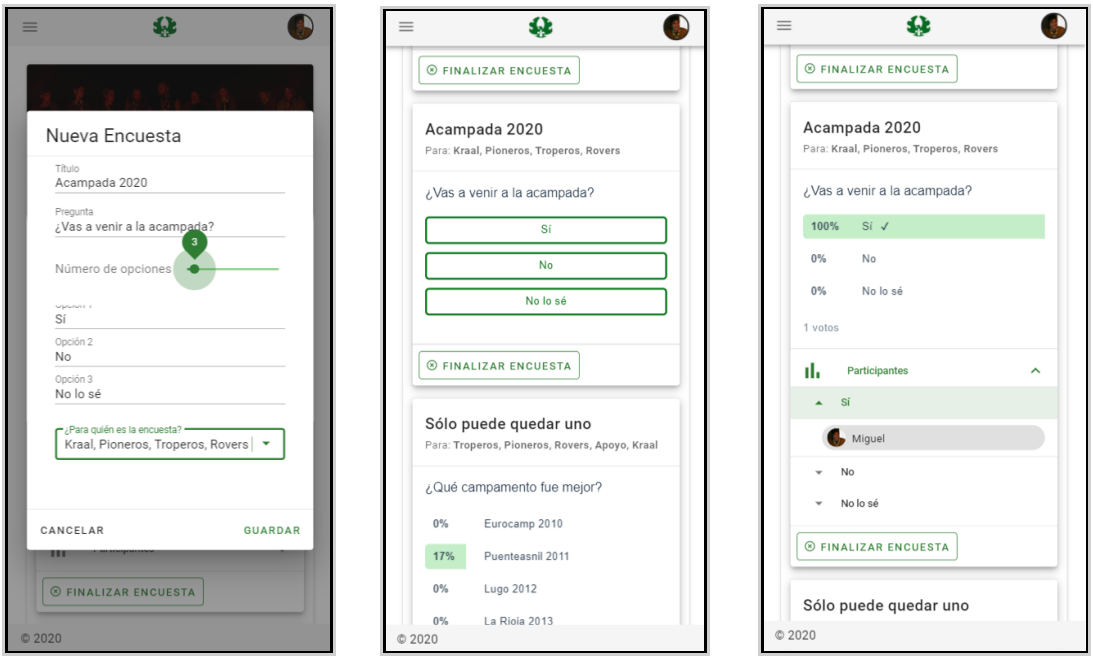

Figura 25: Sección de Encuestas

4.7 Acampadas

Para recopilar la información de las acampadas pasadas y las futuras, se ha usado el componente VCard, que muestra una foto de dicha acampada junto con su información.

Para acceder a cada acampada en concreto, se pulsa el botón entrar, que añade a la ruta *"/acampadas"* el id de la acampada seleccionada. En dicha página los usuarios de tipo familiar y scout pueden ver la información general de la misma junto con el dossier (si ha sido subido). Los miembros de kraal pueden subir un archivo con el dossier, y acceder a otra pestaña con una sección que alberga todos los documentos útiles de las acampadas, que pueden subirse para que el resto de los miembros del kraal los consulten.

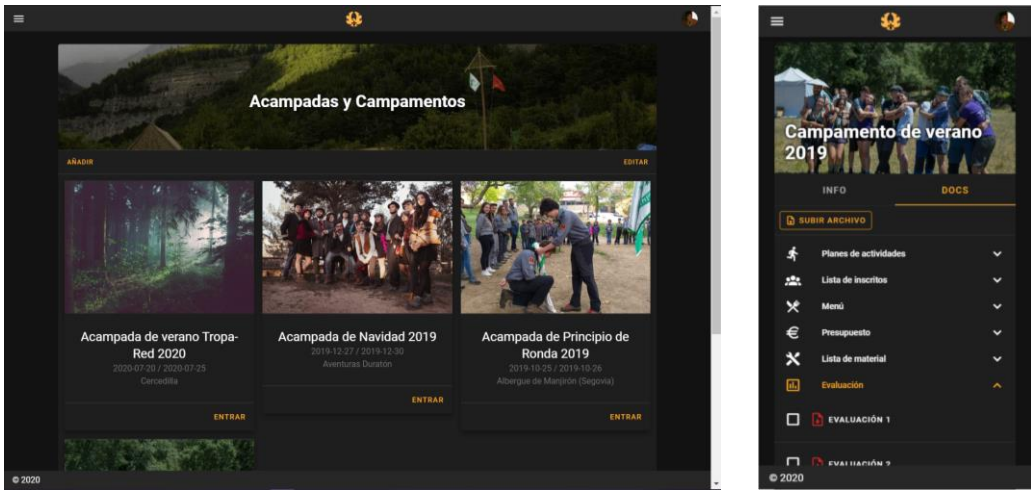

Figura 26: Sección de Acampadas

4.8 Otras secciones

Para concluir, se van a describir las últimas secciones de la aplicación, que cuentan con una funcionalidad similar, centrándose en la colección de contenidos de utilidad.

4.8.1 Rutas

Esta sección recopila información de distintas rutas que ha realizado el grupo scout, de manera muy parecida a la sección de acampadas.

Toda la información es visualizada a través de componentes "<v-card>", mostradas mediante un bucle "v-for", que recorre la información de las rutas leídas del *state* del *store* de Vuex, que previamente las ha obtenido de la base de datos de Firebase.

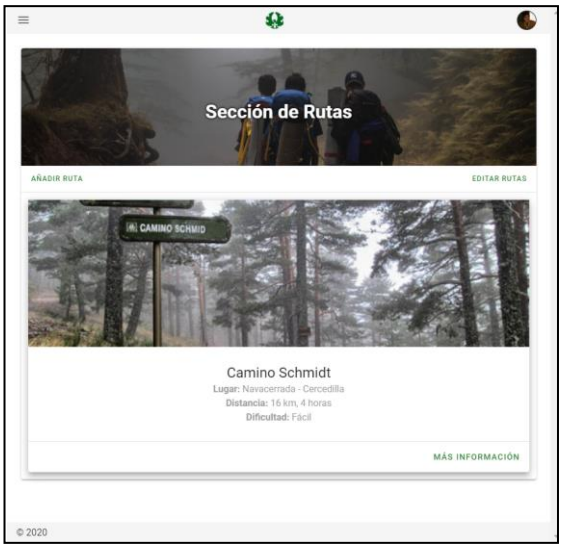

Figura 27: Sección de Rutas

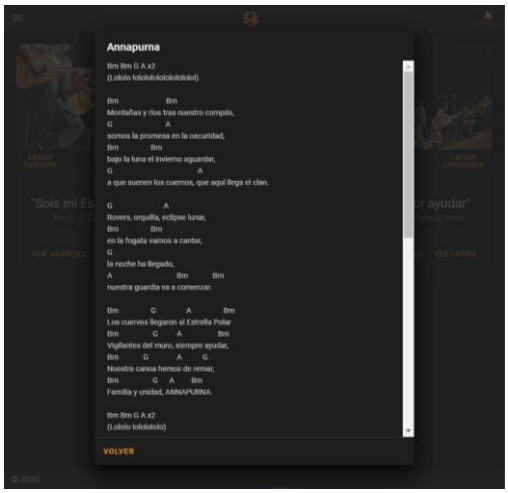

Figura 28: Sección de Música

4.8.2 Música

Este apartado permite añadir las letras y acordes de diferentes canciones populares en el grupo scout.

Para conservar los saltos de línea al introducir las distintas letras en los componentes "<v-text-area>", se ha necesitado modificar dichas cadenas de texto con la función de JavaScript *"replace(/(?:\r\n|\r|\n)/g, '<br/>')"*, que sustituye los saltos de línea por etiquetas HTML "<br/>", que posibilitan la correcta visualización de las letras de las canciones.

4.8.3 Galería

En esta sección se permite la agregación de videos de la plataforma YouTube, para poder visualizarse desde la aplicación.

Para añadir un vídeo, se debe escribir su enlace, que a continuación es modificado para seguir el formato de inserción de vídeos de YouTube, por medio del componente HTML "<iframe>". Estas ventanas con cuadros de reproducción de YouTube están contenidas en componentes "<v-card>".

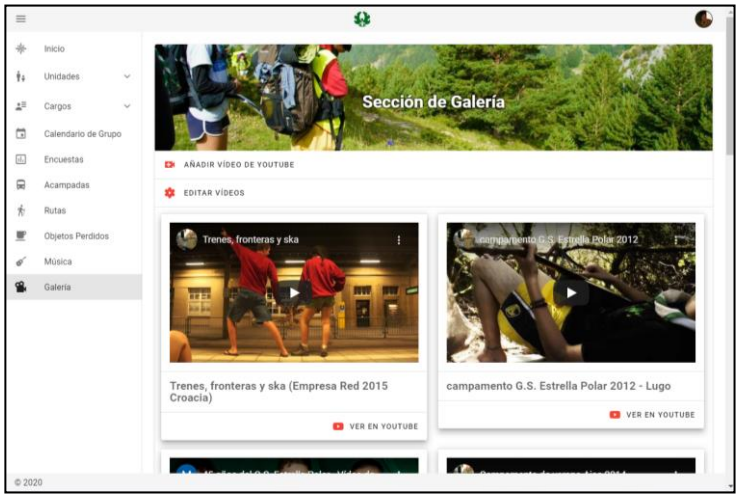

Figura 29: Sección de Galería

Figura 30: Sección de Objetos Perdidos

4.8.4 Objetos perdidos

Tras las actividades o acampadas siempre hay una gran cantidad de objetos de miembros del grupo que se han olvidado o han perdido. En esta sección se pueden añadir objetos acompañados de su foto, para que cualquier usuario que identifique que el objeto es suyo, pueda reclamarlo.

Para ello, se ha insertado un botón "Reclamar", que guarda el id del usuario que lo haya pulsado para así poder mostrar su nombre y su foto en la "<v-card>" del objeto. Esto se ha realizado con el componente "<v-chip>", que, al pulsarlo, accede a la página de perfil del usuario. El objeto puede dejar de ser reclamado por el usuario, o, también, el kraal puede eliminar la reclamación.

4.9 Aspectos adicionales

Asimismo, se han implementado otras funcionalidades entre las que se encuentran:

- La barra superior de la plataforma, "<v-app-bar>", es un componente fijo en toda la aplicación, en la cual se puede acceder a la página de inicio pulsando el logo de Phoenix situado en el centro de la misma, y, a la derecha, muestra la foto de perfil del usuario registrado, en la que se puede seleccionar visitar la página de perfil, cambiar el tema a modo oscuro o claro, y cerrar sesión.

- Snackbars: cada vez que se efectúa un cambio en la aplicación por parte del usuario, aparece en la parte inferior de la pantalla un cuadro de texto que informa de la correcta o incorrecta actualización. Esto se ha realizado mediante un componente programado en el archivo *"snackbar.vue"*.

- Middleware: en la carpeta de Nuxt.js llamada *"middleware"*, se añadió un archivo *"auth.js"* que restringe el acceso a páginas determinadas a los usuarios que no deben poder acceder a ellas. También hace que se impriman por pantalla distintos mensajes según el error que ha sucedido.

- Galería de imágenes de perfil: por medio de un componente se ha desarrollado la opción de poder elegir como foto de perfil una imagen de entre varias proporcionadas por la aplicación de manera estática.

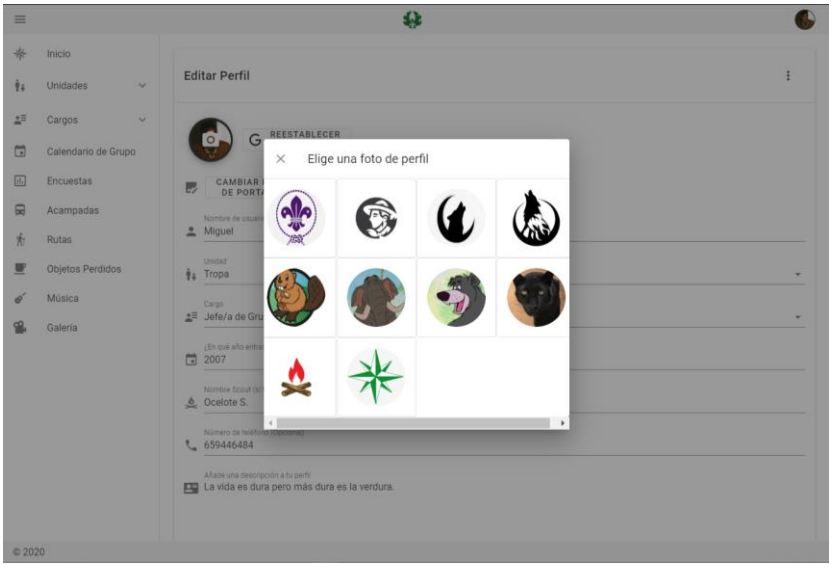

Figura 31: Galería de fotos de perfil

5 Integración, pruebas y resultados

5.1 Pruebas durante el desarrollo

Una gran ventaja de la programación de aplicaciones web, y, más concretamente, en el entorno de desarrollo Nuxt.js, es la capacidad de poder probar cada alteración que se vaya realizando de una manera casi directa, simplemente, al guardar los cambios del proyecto.

Durante el desarrollo del proyecto, se ha ido comprobando el correcto comportamiento de las distintas funcionalidades, de manera que no se ha empezado a programar una funcionalidad distinta hasta que no se ejecutara perfectamente la anterior. Varios métodos útiles para probar el comportamiento del programa han sido, en primer lugar, la consola del navegador, en la que se han ido reflejando las acciones que se llevan a cabo durante la ejecución. En segundo lugar, el uso de la herramienta Vue Devtools de Chrome, que aporta distintas ventajas tales como la posibilidad de comprobar el contenido y los cambios del *store* de Vuex.

En cuanto al aspecto visual de la aplicación, se ha ido verificando en todas las ocasiones posibles, la correcta visualización de todos los elementos necesarios según el tamaño de pantalla indicado. Esto ha sido gracias a la funcionalidad del navegador de poder visualizar la página en distintas proporciones de pantalla, pudiendo elegir hasta el modelo del dispositivo móvil requerido.

5.2 Despliegue de la aplicación

Debido a que el grupo scout Estrella Polar ya disponía de una página web alojada en un servidor en la dirección https://www.gsestrellapolar.es, se ha decidido alojar la aplicación Phoenix en dicho dominio, simplemente añadiendo la ruta "/phoenix" a la página previamente dicha.

Se ha optado por esta opción pudiendo haber elegido otras, como el servicio de hosting gratuito de firebase, debido a que es más sencillo e intuitivo para los miembros del grupo scout acceder directamente desde la página web clásica del grupo.

5.3 Pruebas reales

A comienzos del mes de junio se desplegó la aplicación y dieron comienzo las pruebas con usuarios reales. Estas pruebas se han realizado con los miembros actuales del kraal del grupo scout, cada uno se creó una cuenta, personalizando su perfil y probando las distintas funcionalidades de la aplicación.

Varios problemas que se descubrieron durante este periodo de pruebas incluyeron aspectos como la visualización de imágenes en algunos dispositivos, que se arregló cambiando la etiqueta HTML "<img>" por la de Vuetify "<v-img>", o el hecho de que no aparecía el popup de registro con cuentas de Google, que sucedía porque el navegador tenía bloqueado en ese sitio web el uso de ventanas emergentes.

Junto con las instrucciones de la aplicación, se envió una pequeña encuesta para que describieran la experiencia con el primer uso de la aplicación (anexo B).

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Este proyecto ha sido elaborado con la motivación clara de poder aunar toda la información aprendida durante el grado de ingeniería informática, y usarla para mejorar el funcionamiento del grupo scout Estrella Polar.

Durante el desarrollo del trabajo, se ha realizado una tarea de investigación considerable, puesto que la mayor parte de los contenidos relacionados con el proyecto no se incluyen en las asignaturas vistas en el grado. Esto ha supuesto un gran ejercicio de persistencia en recopilar la información adecuada, así como de diseñar e implementar la aplicación de manera correcta, para evitar los errores trabajados durante los años de carrera.

En cuanto a los conocimientos aprendidos durante el progreso del proyecto se encuentran las nociones del panorama actual en los ámbitos de frontend y backend en aplicaciones web, y, sobre todo, el hecho de poder afrontar un proyecto desde cero usando dichos conocimientos.

La valoración de la aplicación desarrollada es muy positiva, puesto que se han conseguido implementar los requisitos funcionales de manera correcta, y los usuarios externos que han podido probar sus distintas funcionalidades han considerado que es una plataforma verdaderamente beneficiosa para las necesidades del grupo scout.

En definitiva, la creación de la aplicación web Phoenix, ha servido para mejorar y favorecer la labor llevada a cabo por los grupos scout, un método de educación que sin duda genera un impacto en la sociedad. Poder usar la tecnología como un instrumento para promover la transmisión de valores esenciales hoy en día a la juventud, de manera desinteresada, es sin duda una razón muy positiva por la que realizar este tipo de proyectos.

6.2 Trabajo futuro

La aplicación creada durante este proyecto se ha desarrollado para poder implantarse de manera fija en el método de trabajo de la gestión del grupo scout, por lo que seguirá cambiando y adecuándose a las necesidades del grupo.

Las principales funcionalidades que se quieren implementar en el futuro incluyen aspectos como:

- Inscripciones de secretaría para miembros del grupo, elaborando un formulario que los usuarios deban rellenar para así guardar toda la información burocrática necesaria de cada miembro en la plataforma.

- Posibilidad para los usuarios de las distintas unidades de crear “posts” y publicarlos en su sección de unidades.

- Añadir un foro para que las familias puedan preguntar directamente a los monitores las dudas que tengan.

- Uso de “notificaciones push” que informen a los usuarios de los distintos cambios que se producen en la aplicación.

- Método para poder mantener la sesión de usuario iniciada, aunque se salga de la aplicación, teniendo así también que cambiar la política de privacidad, debido al uso de “cookies”.

Referencias

[1] *Baden-Powell*, World Scout Bureau Global Support Centre, Kuala Lumpur World Organization of the Scout Movement. Consultado en: junio de 2020. [Online]. Disponible: https://www.scout.org/es/node/9694

[2] *¿Cuántos scouts hay en el mundo?*, OMMS. Consultado en: junio de 2020. [Online]. Disponible: https://scouts.es/cuantos-scouts-hay-en-el-mundo/#:~:text=Seg%C3%BAn%20fuentes%20de%20la%20OMMS,en%20163%20pa%C3%ADses%20del%20mundo.

[3] *Benefits of Native Mobile App Development for Businesses*, NewAgeSys Inc., 12 March, 2019. Consultado en: junio de 2020. [Online]. Disponible: https://www.newagesmb.com/blog/benefits-native-mobile-app-development

[4] Jobe, W., *Native Apps Vs. Mobile Web Apps*, International Journal of Interactive Mobile Technologies (iJIM). Consultado en: junio de 2020. [Online]. Disponible: https://online-journals.org/index.php/i-jim/article/view/3226/2840

[5] Ivano Malavolta, *Beyond native apps: web technologies to the rescue! (keynote)*, Vrije Universiteit Amsterdam, The Netherlands, 2016. Consultado en: junio de 2020. [Online]. Disponible:https://www.researchgate.net/publication/309450985_Beyond_native_apps_web_technologies_to_the_rescue_keynote

[6] Margaret Rouse, *Web application (Web app)*, August 2019. Consultado en: junio de 2020. [Online]. Disponible: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app

[7] *¿Qué es JavaScript?*, MDN web docs. Consultado en: junio de 2020. [Online]. Disponible:https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript

[8] *Ayax Introduction*, W3 schools. Consultado en: junio de 2020. [Online]. Disponible: https://www.w3schools.com/xml/ajax_intro.asp

[9] Oleg Uryutin, *A brief history of web app*, September 13, 2018. Consultado en: junio de 2020. [Online]. Disponible: https://medium.com/@aplextor/a-brief-history-of-web-app-50d188f30d

[10] Tandel, Sayali & Jamadar, Abhishek, *Impact of Progressive Web Apps on Web App Development*, 2018. Consultado en: junio de 2020. [Online]. Disponible:https://www.researchgate.net/publication/330834334_Impact_of_Progressive_Web_Apps_on_Web_App_Development

[11] Tim A. Majchrzak, Andreas Biørn-Hansen and Tor-Morten Grønli, *Progressive Web Apps: the Definite Approach to Cross-Platform Development?*, 2018. Consultado en: junio de 2020. [Online].
Disponible: https://core.ac.uk/reader/143481552

[12] Madhuri A. Jadhav, Balkrishna R. Sawant, Anushree Deshmukh, *Single Page Application using AngularJS*, 2015. Consultado en: junio de 2020. [Online].
Disponible:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.736.4771&rep=rep1&type=pdf

[13] Vue.js. Consultado en: junio de 2020. [Online].
Disponible: https://vuejs.org/v2/guide/index.html

[14] React.js. Consultado en: junio de 2020. [Online].
Disponible: https://reactjs.org/docs/getting-started.html

[15] Angular. Consultado en: junio de 2020. [Online].
Disponible: https://angular.io/docs

[16] Kin Lane, *Overview Of The Backend as a Service (BaaS) Space*, May 2013. Consultado en: junio de 2020. [Online].
Disponible: http://kinlane-productions.s3.amazonaws.com/whitepapers/API+Evangelist+-+Overview+of+the+Backend+as+a+Service+Space.pdf

[17] Adrián Alonso Vega, *Backend as a Service o como prescindir de un backend developer*, April 13, 2017. Consultado en: junio de 2020. [Online].
Disponible: https://medium.com/codenares/backend-as-a-service-o-como-prescindir-de-un-backend-developer-facdde6e3132

[18] Node.js. Consultado en: junio de 2020. [Online].
Disponible: https://nodejs.org/es/

[19] Npm. Consultado en: junio de 2020. [Online].
Disponible: https://www.npmjs.com/

[20] Nuxt.js. Consultado en: junio de 2020. [Online].
Disponible: https://nuxtjs.org/guide

[21] Vuetify. Consultado en: junio de 2020. [Online].
Disponible: https://vuetifyjs.com/en/introduction/why-vuetify/

[22] Chunnu Khawas, Pritam Shah, *Application of Firebase in Android App Development-A Study*, June 2018. Consultado en: junio de 2020. [Online].
Disponible:https://www.researchgate.net/profile/Chunnu_Khawas/publication/325791990_Application_of_Firebase_in_Android_App_Development-A_Study/links/5bab55ed45851574f7e6801e/Application-of-Firebase-in-Android-App-Development-A-Study.pdf

[23] *Elige una base de datos: Cloud Firestore o Realtime Database*, Firebase. Consultado en: junio de 2020. [Online].
Disponible: https://firebase.google.com/docs/database/rtdb-vs-firestore?hl=es

[24] *Planes de precios*, Firebase. Consultado en: junio de 2020. [Online].
Disponible: https://firebase.google.com/pricing?authuser=0

[25] Vuex. Consultado en: junio de 2020. [Online].
Disponible: https://vuex.vuejs.org/

[26] Prokopis Pietris. *Vue-Poll*. Consultado en: junio de 2020. [Online].
Disponible: https://madewithvuejs.com/vue-poll

Glosario

API	Application Programming Interface
PWA	Progressive Web Application
SPA	Single Page Application
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
XML	Extensible Markup Language
JSON	JavaScript Object Notation
URL	Uniform Resource Locator
NPM	Node Package Manager
PHP	Hypertext Pre-Processor
SQL	Structured Query Language
BaaS	Backend as a Service
DOM	Document Object Model
MVC	Model View Controller
MVVM	Model View View-Model

Anexos

A Diagrama de Casos de Uso

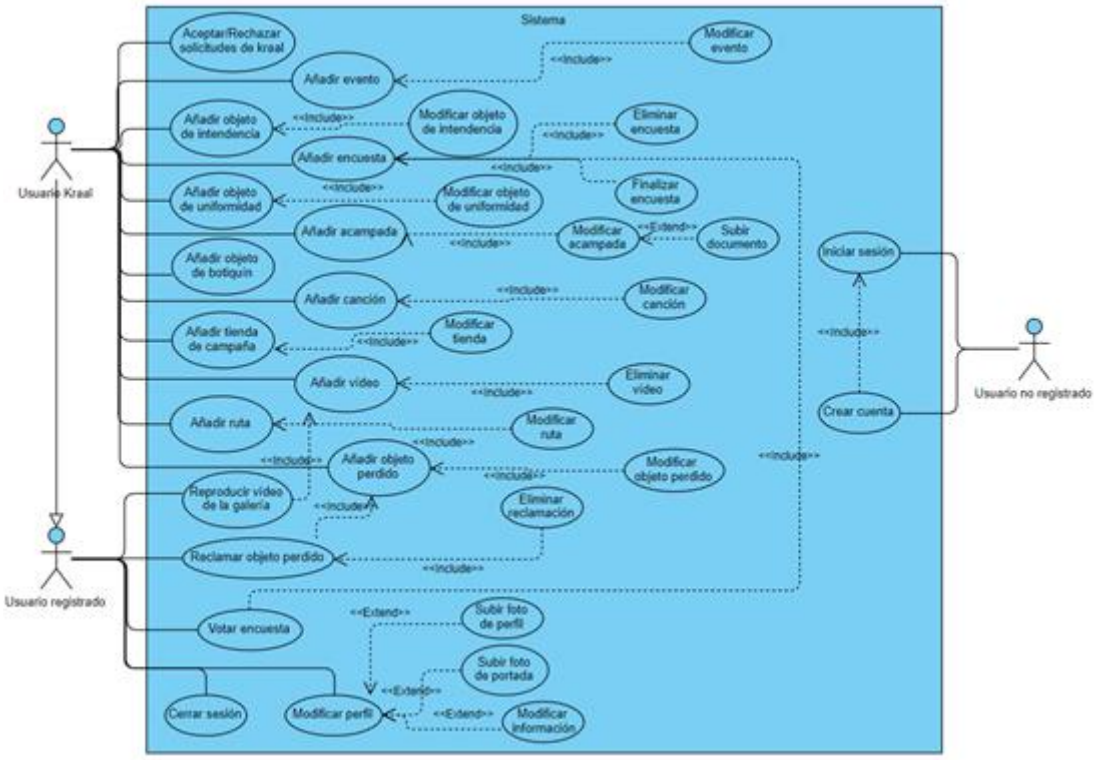

Figura 32: Diagrama de Casos de Uso

B Resultados de la encuesta realizada a usuarios reales

- Formulario de la encuesta:

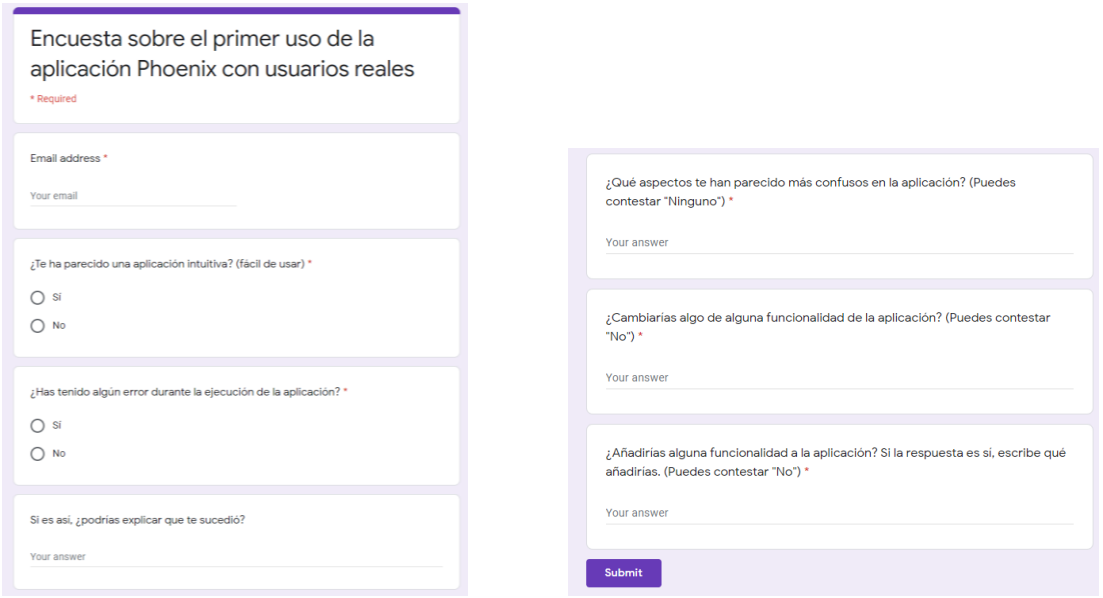

Figura 33: Formulario de la encuesta a usuarios

- Respuestas a la encuesta:

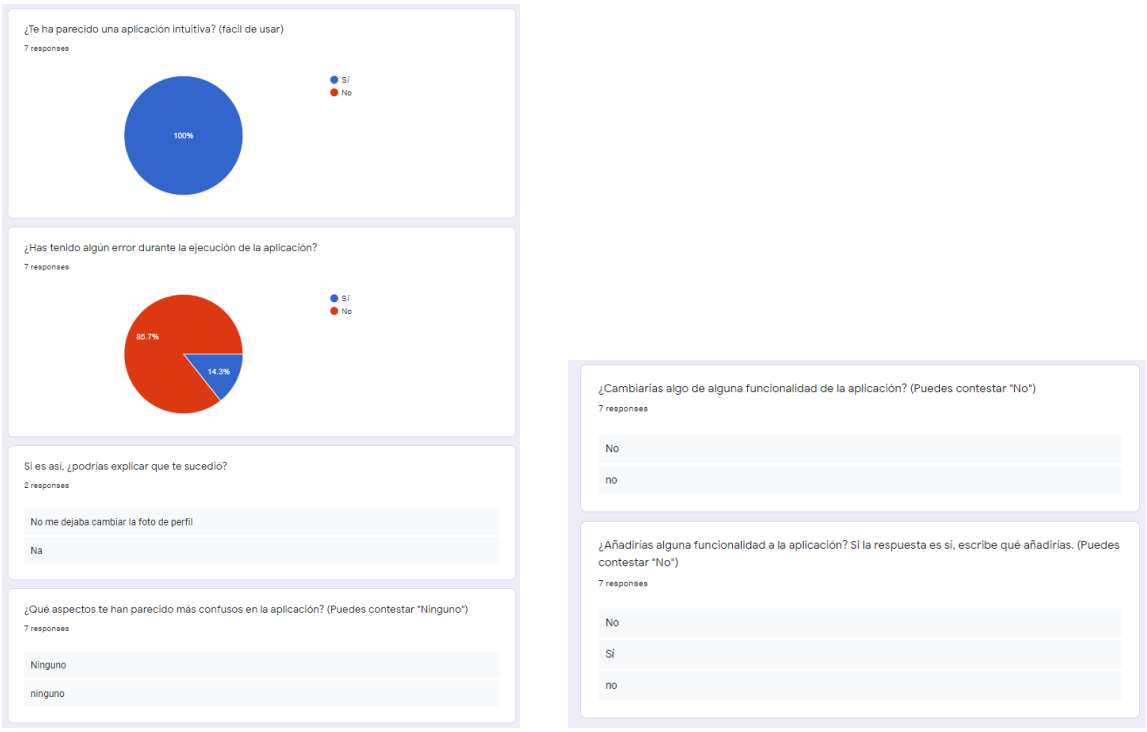

Figura 34: Respuestas a la encuesta a usuarios